UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Herramientas de apoyo a la generación de rutas óptimas para vehículos submarinos autónomos

Cristian David Estupiñán Ojeda

Tutor: **Dr. D. José Daniel Hernández Sosa**

Trabajo Fin de Grado

Grado en Ingeniería Informática

Universidad de Las Palmas de Gran Canaria
29 de mayo de 2018

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

TFT04

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D/Dª Cristian David Estupiñán Ojeda, autor del Trabajo de Fin de Título Herramientas de apoyo a la generación de rutas óptimas para vehículos submarinos autónomos,
correspondiente a la titulación Grado en Ingeniería Informática,
en colaboración con la empresa/proyecto (indicar en su caso) Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería.

S O L I C I T A

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

[ ] Se ha iniciado o hay intención de iniciarlo (defensa no pública).
[x] No está previsto.

Y para que así conste firma la presente.

Las Palmas de Gran Canaria, a 30 de Mayo de 2018.

El estudiante

Fdo.: ESTUPIÑAN OJEDA CRISTIAN DAVID - 54149849Y

Digitally signed by ESTUPIÑAN OJEDA CRISTIAN DAVID - 54149849Y
Date: 2018.05.30 19:58:56 +01'00'

A rellenar y firmar **obligatoriamente** por el/los tutor/es

En relación a la presente solicitud, se informa:

[x] Positivamente

[ ] Negativamente
(la justificación en caso de informe negativo deberá incluirse en el TFT05)

Fdo.: HERNANDEZ SOSA JOSE DANIEL - 42171274S

Firmado digitalmente por HERNANDEZ SOSA JOSE DANIEL - 42171274S
Fecha: 2018.05.30 20:43:36 +01'00'

DIRECTOR DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

"A menudo, las personas que son exitosas aman lo que hicieron para poder perseverar cuando las cosas se ponen difíciles."
Steve Jobs.

Agradecimientos

Quiero agradecer a mi tutor académico, José Daniel Hernández Sosa, tanto por guiarme en este proyecto como por valorar el trabajo que he ido desempeñando. Asimismo, agradezco el apoyo de mis compañeros y amigos, quienes siempre han estado ahí para ayudarme cuando lo necesitaba, así como a mi familia que tanto económica como emocionalmente han ayudado en lo posible a que pueda cumplir mis sueños.

Resumen

En este proyecto se implementan varias herramientas de apoyo a la generación de rutas óptimas para vehículos submarinos autónomos. Esta implementación se ha realizado tomando como origen herramientas desarrolladas por el equipo de investigación en Robótica y Oceanografía Computacional (ROC) del Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería, adaptándolas a OCTAVE, un software que cuenta con licencia libre de uso (GNU GPL). Asimismo, se han incorporado nuevas herramientas y soluciones de estructuración de datos con el mismo fin de proporcionar un amplio abanico de herramientas gratuitas, fomentando así el uso del software libre y su potencialidad en el ámbito de la investigación y del desarrollo de aplicaciones científicas.

PALABRAS CLAVE: octave, matlab, gnu, gpl, software libre, herramientas, rutas, submarinas, vehículos autónomos

Abstract

In this project several support tools are implemented to generate optimal routes for autonomous underwater vehicles. This implementation has been carried out using tools created by the Robotics and Computational Oceanography (ROC) research team from the University Institute of Intelligent Systems and Numeric Applications in Engineering (IUSIANI). Those tools were adapted to OCTAVE, a software that has a free license (GNU GPL). Likewise, new tools and data structuring solutions have been incorporated with the same purpose of providing a wide range of free tools, thus promoting the use of free software and its potential in the field of research and development of scientific applications.

KEY WORDS: octave, matlab, gnu, gpl, free software, tools, routes, submarines, autonomous vehicles

Índice general

Índice de figuras

Listings

Acrónimos

MATLAB: Herramienta de software matemático
OCTAVE: Herramienta de software matemático libre
NETCDF: Network Common Data Form
CMEMS: Copernicus Marine Environment Monitoring Service
IBI: Iberia-Biscay-Ireland Regional Seas
PDE: Institución Puertos del Estado
IUSIANI: Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería
PLOT: Representación gráfica
CESGA: Centro de Supercomputación de Galicia
HYCOM: HYbrid Coordinate Ocean Model
WYSIWYG: Lo que ves es lo que obtienes
POLLING: Consulta mediante encuestas

Capítulo 1

Introducción

El software libre se considera como tal si cumple con cuatro libertades principales [10]:

- **1**: Libertad de poder ejecutar el programa para cualquier fin.
- **2**: Libertad de estudiar su funcionamiento y adaptación a lo que uno desee hacer con él.
- **3**: Libertad para distribuir copias que ayuden a tus vecinos.
- **4**: Libertad para distribuir tus propias copias modificadas o mejoradas.

Todas y cada una de estas libertades para un equipo de investigación permiten que puedan generar funcionalidades y distribuirlas sin restricciones del proveedor, así como por un lado, poder realizar las modificaciones necesarias al software, y por otro, evitar las exclusividades del software.

GNU Octave es el software y lenguaje de alto nivel utilizado concretamente para la realización de este trabajo, empleado con el fin de realizar operaciones numéricas computacionales complejas, proporcionando un amplio conjunto de herramientas y funciones para poder resolver la gran mayoría de problemas numéricos tanto lineales como no lineales [8].

Uno de los puntos fuertes de Octave radica en su versatilidad, pues permite resolver problemas algebraicos comunes, integraciones numéricas, manipulación de polinomios, ecuaciones diferenciales, y un largo etcétera de este tipo de problemas. Asimismo, posee funciones dedicadas al tratamiento de cadenas de caracteres, e integraciones con otro tipo de lenguajes, como por ejemplo con Java. Y, finalmente, se puede destacar su capacidad de personalización mediante funciones definidas por el propio usuario y el uso de paquetes externos basados tanto en C, C++ y Fortran.

Por otro lado, existe MATLAB, un software dedicado a realizar también operaciones numéricas que posee bastantes similitudes con Octave, tanto con el tipo de problemas que resuelve como con la mayoría de las funciones predefinidas que utiliza. Sin embargo, existe una gran diferencia con respecto a Octave en el tipo de licencia que ambos software utilizan. Octave emplea un tipo de licencia libre (GNU GPL), al contrario que MATLAB.

1.1. Estructura del documento

En este documento, comenzaré con el estado actual, la motivación y los objetivos, de igual manera, resaltaré las competencias adquiridas y las herramientas que se han utilizado en este proyecto. Luego, expondré los recursos que he utilizado para su realización y planificación. Seguidamente, detallaré cada una de las partes principales del trabajo en los sucesivos capítulos y los resultados que se han ido obteniendo. Finalmente, el documento contará con las conclusiones obtenidas, los apéndices y la bibliografía.

Capítulo 2

Estado actual del tema

En este segundo capítulo abordaré el estudio teórico del procesamiento de archivos NETCDF utilizado en la implementación de las diferentes herramientas del proyecto, como también el servicio de monitorización del medio ambiente marino Copernicus.

2.1. Estado de las herramientas actualmente desarrolladas

En la actualidad, el grupo de investigación en Robótica y Oceanografía Computacional (ROC) del Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (IUSIANI) dispone de un conjunto de herramientas y utilidades que se han ido desarrollando en el marco de diferentes proyectos y publicaciones.

Estas aplicaciones no están adecuadamente integradas, además de no encontrarse desarrolladas con software libre.

Por otro lado, existe una dificultad para que los resultados de un determinado trabajo puedan volver a generarse en el caso de que así se requiera en el futuro, tales como: Revisión de artículos, continuación de una línea de investigación, cooperación con otros grupos, etc.

Es por ello por lo que se hace necesario almacenar adecuadamente la información relativa a los datos de entrada utilizados, así como la configuración de las misiones, la parametrización del simulador, el tipo de vehículos y el resto de elementos que intervengan.

2.2. Introducción a los archivos netCDF

Los archivos netCDF, o por sus siglas (Network Common Data Form) corresponden a una interfaz dedicada a una librería de funciones para almacenar y poder recuperar de nuevo datos en forma de matrices. Estas matrices poseen una estructura rectangular n-dimensional que contiene elementos del mismo tipo de datos.

"NetCDF es una abstracción que admite una vista de datos como una colección de objetos autodescriptivos y portátiles a los que se puede acceder a través de una interfaz simple"[13]. Esto quiere decir que se puede acceder a los valores que se tengan de la matriz n-dimensional de manera directa, existiendo una serie de utilidades genéricas de las cuales trataré en profundidad en los siguientes capítulos, éstas permiten poder transformar, combinar, analizar y mostrar los campos especificados de los datos que se deseen tratar.

Netcdf soporta interfaces para diferente tipos de lenguajes, tanto para C, C++ y Fortran. En este caso ya que se trabajará con una librería externa para el software de Octave, se tomará una versión basada en el lenguaje C++, en concreto netCDF-CXX en su versión 4.2. También sería posible poder instalar una versión basada en Fortran, sin embargo por problemas de compatibilidades con el entorno de desarrollo es preferible utilizar la versión de C++.

2.2.1. Arquitectura netCDF

En la figura 2.1 se tiene un ejemplo claro de la arquitectura en la que se basa netCDF tomada de la página oficial del desarrollador:

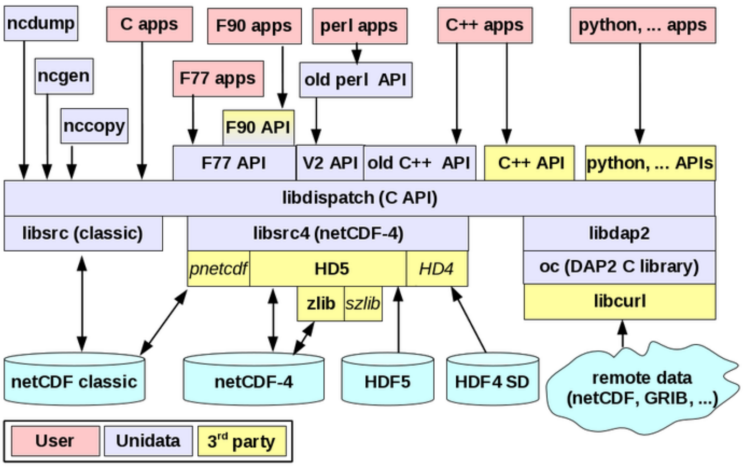

Figura 2.1: Arquitectura NETCDF

De esta imagen se puede observar claramente que la mayoría de APIs desarrolladas para lenguajes como Python, Ruby, Matlab, R, entre otros se basan en la realizada en C.

2.2.2. Rendimiento de netCDF

Uno de los objetivos principales de netCDF consiste en permitir accesos a pequeños subconjuntos de grandes datasets de manera eficiente usando para ello acceso directo en lugar de acceso secuencial. Gracias a esto, se puede acceder de manera más eficiente cuando se leen datos que difieren del orden en el que se han escrito, o cuando se deben leer en órdenes diferentes según las aplicaciones finales [13].

2.2.3. Tipos de datos de netCDF

Los tipos de datos atómicos que se utilizan en cualquier interfaz netCDF son los siguientes [13]:

- **NCBYTE**: 8-bit signed integer
- **NCUBYTE**: 8-bit unsigned integer
- **NCCHAR**: 8-bit character byte
- **NCSHORT** 16-bit signed integer
- **NCUSHORT** 16-bit unsigned integer
- **NCINT** (or NCLONG) 32-bit signed integer
- **NCUINT** 32-bit unsigned integer
- **NCINT64** 64-bit signed integer
- **NCUINT64** 64-bit unsigned integer
- **NCFLOAT** 32-bit floating point
- **NCDOUBLE** 64-bit floating point
- **NCSTRING** variable length character string

Cabe destacar que el único tipo de estructura de datos que soporta la estructura clásica de netCDF es la colección de arrays con atributos de vectores, es decir, no sería útil para el almacenamiento de listas encadenadas, árboles, matrices dispersas y otro tipo de estructuras que requieran el uso de punteros. En el caso de la versión 4 de netCDF que es la que se utilizará en este proyecto, se incorpora de manera nativa los arrays de longitud variable [11].

2.2.4. Modelo de datos de netCDF

Con respecto al modelo de datos en el que se basa netCDF, se debe tener en cuenta que contiene lo siguiente: dimensiones, variables y atributos, cada uno de ellos con un ID que los identifica.

En concreto, los archivos creados con la versión 4 de este formato utilizado en el proyecto permite además la representación de grupos, utilizados para organizar grandes números de variables [12]. Ver figura 2.2

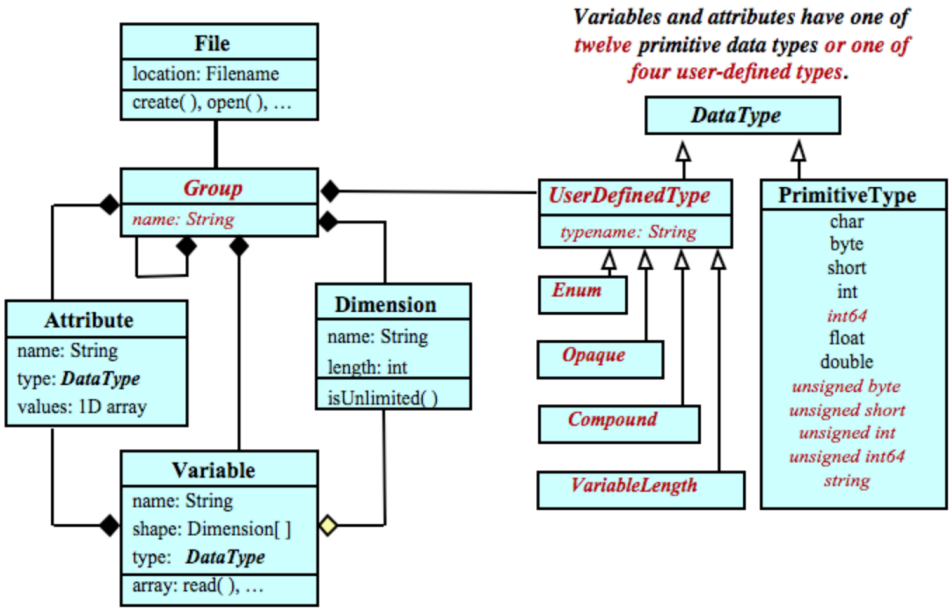

Figura 2.2: Modelo de datos NETCDF

Cada grupo permitiría incluir a su vez un nuevo dataset netCDF, con sus propias dimensiones, variables y atributos, a la vez que otros subgrupos.

Dimensiones

Las dimensiones se suelen utilizar para representar dimensiones físicas reales como puede ser el tiempo, la latitud, la longitud, la profundidad, etcétera. Este tipo de dimensiones son las que utiliza el servicio de monitoreo del medio ambiente marino Copernicus, del cual hablaré a continuación.

Una dimensión contiene pues un nombre y una longitud, basándonos en la figura anterior 2.2.

Con respecto a la longitud de la dimensión, puede ser un número entero positivo. Se debe tener en cuenta que en la versión 4 de netCDF se permite el uso de dimensiones ilimitadas. Sin embargo en la versión clásica solo se permite que exista una dimensión ilimitada. [12].

Variables

Las variables en netCDF se utilizan para poder almacenar la mayor parte de los datos en un modelo netCDF. Representa pues una matriz de valores que deberán ser del mismo tipo. Una variable debe de tener un nombre, el tipo de datos asociado a la misma y un factor de forma asociado al tipo de dimensiones que se le especifique que debe utilizar. Por otro lado, una variable también puede tener, o no, atributos asociados, los cuales se podrán añadir, eliminar o cambiar a posteriori de la creación de la variable. Ver la figura 2.2

Atributos

Con respecto a los atributos, estos son utilizados para guardar los metadatos de los datos almacenados, proporcionando información sobre una variable específica e identificándose por un nombre o identificador de variable.

Algunos atributos a su vez incluyen información sobre el dataset y se denominan como atributos globales, identificados por el nombre de atributo seguido de un nombre de variable nulo para los formatos netCDF basados en C, C++ o Fortran [12].

2.3. Servicio Copernicus CMEMS

El servicio de monitoreo del medio ambiente marino Copernicus CMEMS, es un servicio que proporciona información sistemática y regular sobre determinados estados físicos, variabilidades y dinámicas de los ecosistemas marinos para los océanos del mundo y los mares regionales europeos.

Las observaciones y predicciones producidas por el servicio soportan todo tipo de aplicaciones marinas, incluyendo las siguientes:

- Seguridad marina
- Recursos marinos
- Ambiente marino y costero
- Tiempo, predicciones por temporadas y climatologías.

Gracias al conjunto de datos que proporciona el CMEMS sobre diferentes tipos de variables marinas, se pueden considerar una gran cantidad de casos de uso, incluyendo también el enfocado a este proyecto: rutas marinas para vehículos autónomos submarinos. En la figura 2.3 se tiene un ejemplo obtenido de Copernicus [1]

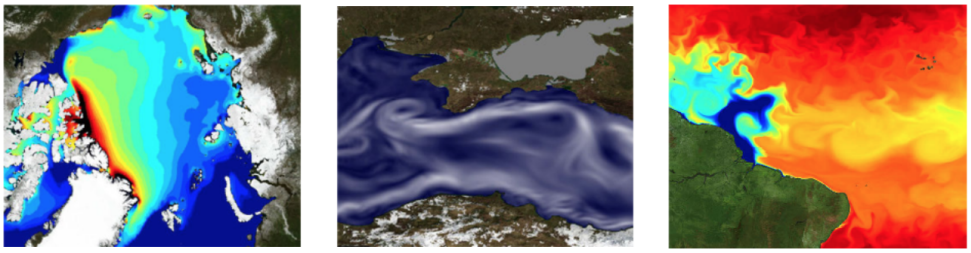

Figura 2.3: Mapas sobre distintas variables obtenidas por CMEMS

Cabe destacar que para poder hacer uso de sus servicios es necesario un registro previo en su página web asociada. Esta información se ha obtenido a partir de su página oficial [1].

2.3.1. Dominio regional, región IBI

Con respecto a los dominios regionales que abarca CMEMS, se tienen los siguientes:

- Océano global
- Océano ártico
- Mar báltico
- Mares europeos de la plataforma noroeste

- Mar regional de Iberia-Vizcaya-Irlanda
- Mar mediterráneo
- Mar negro

En este proyecto se trabajará concretamente sobre el mar regional de Iberia-Vizcaya-Irlanda. Ver figura 2.4 .

El análisis oceanográfico y de predicción de este área, lo lleva a cabo diariamente la institución Puertos del Estado (PdE), quien proporciona predicciones de 5 días sobre la hidrodinámica, incluyendo procesos de alta importancia realizados con frecuencia para caracterizar los determinados procesos de escala marina de dicha región (por ejemplo: Fuerza de marea, descarga de ríos de agua dulce, presión atmosférica). Así como una actualización semanal a baja escala del análisis de la región IBI y su respectivo histórico [3].

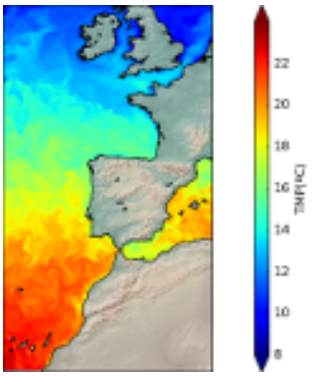

Figura 2.4: Mar regional de Iberia-Vizcaya-Irlanda

En relación a la profundidad que se producen en los modelos generados por PdE, existen un total de 50 niveles que abarcan desde los 5800 metros de profundidad hasta la superficie, siendo el grosor de cada nivel o capa de en torno a 1 metro de resolución cerca de la superficie y de 400 metros en los niveles más profundos.

Al obtener un modelo cualquiera de esta región mediante CMEMS, se obtiene un fichero netCDF de la siguiente forma:

CMEMS_fileVersion_region_PHY_NRT_PdE_freqFlag _validDate_valiDate_RbulletinDate_productType.nc

Teniendo cada uno de los campos del fichero el siguiente significado:

- **region**: Corresponde a un código de 3 letras de la región, IBI en este caso.

- **fileVersion**: Corresponde con la versión del archivo.

- **freqFlag**: Corresponde con la frecuencia de los valores de datos en el archivo, siendo 01hav datos tomados por horas y 01dav datos tomados diariamente.

- **validDate**: Corresponde con la fecha válida de los campos contenidos en el archivo.

- **bulletinDate**: Corresponde con la fecha en la que los datos se han producido.

- **productType**: Corresponde con un código de dos letras para el tipo de producto.

Cada uno de los presentes datos son proporcionados en el documento [3].

Hay que tener en cuenta que este modelo es el proporcionado por Puertos del Estado, internamente en el proyecto trabajaré con el esquema propuesto por la división de Robótica y Oceanografía Computacional del IUSIANI (Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería), aunque también proporcionaré un esquema propio en los sucesivos capítulos del presente documento.

2.3.2. Dominio temporal

El dominio temporal que permite consultar el motor del CMEMS abarca desde el 1 de enero de 1992 hasta la actualidad. Para cada modelo que se obtenga de las consultas, existen dos clases de dominios temporales.

Un modelo donde se obtienen los datos por horas, conocido como 01hav, y otro tipo de modelos donde los datos se recogen por días enteros, permitiendo escoger al usuario el intervalo de tiempo que desee entre dichos márgenes.

El esquema de predicción de los diferentes modelos, concretamente de la institución Puertos del Estado se muestra en la figura siguiente 2.5:

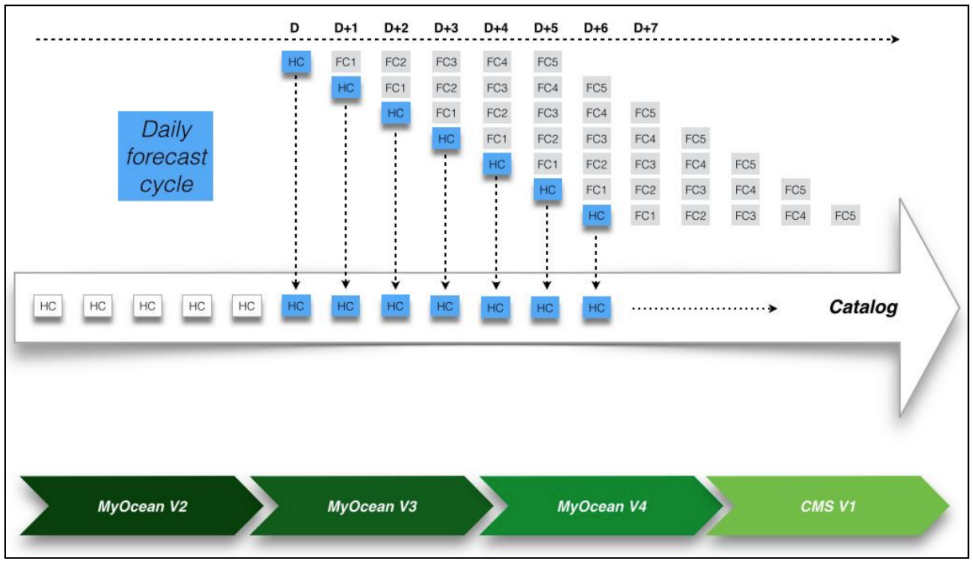

Figura 2.5: Esquema de predicciones de Puertos del Estado

2.3.3. Parámetros oceánicos

Con respecto a los parámetros oceánicos, estos varian con respecto a las diferentes versiones de los archivos que se publican cada cierto tiempo, el glosario de parámetros/variables de los archivos netCDF actuales se encuentra en el siguiente enlace [2].

Algunos de los más importantes y usados por las herramientas son los siguientes:

- **T**: Temperatura.
- **SST**: Temperatura de la superficie del agua.
- **UV**: Velocidad de la corriente.
- **WIND**: Viento.

Todas las variables presentes en el glosario pertenecen a tres grandes grupos: variables físicas, variables de olas y variables biológicas.

2.3.4. Obtención de datos de modelos

Para obtener los modelos que se deseen investigar, es necesario estar registrado en el servicio CMEMS. Una vez se encuentre registrado, se deberán tener en cuenta cada una de las secciones anteriormente mencionadas.

Por un lado la región sobre la que tomar los datos, en el proyecto será la correspondiente a la región IBI. Por otro lado se deben marcar las variables/parámetros que contendrán los datasets, y finalmente habrá que definir el intervalo de tiempo del mismo.

Tras definir todos estos campos, el motor de búsqueda de Copernicus proporcionará al usuario un modelo de uno o varios datasets cumpliendo cada una de estas características indicadas.

2.3.5. Posibles casos de uso

Algunos de los posibles casos de uso del servicio al suministrar datos sobre corrientes, vientos y hielo marino, entre otros, pueden ser para poder mejorar los servicios de enrutamiento de buques, operaciones en alta mar de búsqueda y rescate, o en el caso del propio proyecto, la creación de herramientas para realizar rutas marinas para vehículos autónomos submarinos.

Capítulo 3

Motivación y objetivos

En este capítulo explicaré que motivación he tenido para seleccionar este proyecto y cuales son los objetivos y competencias básicas que se cumplen con la realización del mismo.

3.1. Motivación

Uno de los factores que han evocado en mí una gran motivación a la hora de realizar este trabajo ha sido el de poder diseñar y adaptar herramientas para una de las líneas de investigación actuales de la División de Robótica y Oceanografía Computacional, que forma parte del Instituto de Sistemas Inteligentes y Aplicaciones Numéricas en la Ingeniería.

Se parte de vehículos autónomos que tienen como uno de sus principales objetivos la maximización de la persistencia, al objeto de permitirles ejecutar misiones cada vez más prolongadas en el tiempo, donde es importante el rango de actuación que pueden alcanzar en tareas de inspección y muestreo.

La planificación de rutas se convierte pues en una tarea crucial para que los vehículos sean realmente efectivos, puesto que las condiciones ambientales tienen un gran impacto en su comportamiento. Es interesante, por tanto, disponer de herramientas de apoyo a la planificación de rutas que faciliten su ejecución y evaluación. Siendo estas herramientas y su desarrollo el principal motivo por el que realizo este proyecto, al tratarse de un tema íntegramente relacionado con la mención que curso (Computación).

3.2. Objetivos principales

Los objetivos principales de este proyecto son los siguientes:

- Por un lado, se pretende disponer de componentes de obtención de datos que permitan el acceso a servidores de predicciones oceanográficas, siendo posible especificar parámetros tales como la región, intervalo temporal, la resolución y las variables de interés.
- Por otra parte, es necesario disponer de visualizadores en tres dimensiones para la inspección de las salidas de los diferentes modelos.
- En siguiente lugar, se realizará la estructuración de los datos presentes actualmente en la división ROC-IUSIANI.
- Por último, se pretende el desarrollo de componentes de visualización para los datos de navegación del vehículo, tanto desde simuladores como a partir de datos reales.

3.3. Competencias específicas adquiridas

Las principales competencias cubiertas en el desarrollo del proyecto son las siguientes:

- **CP03**: Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.

  Esta capacidad se ha obtenido al estructurar los datos presentes en el sistema de archivos de modelos de regiones del IUSIANI y al presentar el sistema de nombres para un mejor indexado de los archivos netCDF.

- **CP04**: Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.

  A la hora de diseñar la estructura de datos del sistema he obtenido esta capacidad al aplicar ciertas técnicas de estructuración y analizar cuál es la idónea en el proyecto.

- **CP06**: Capacidad para desarrollar y evaluar sistemas interactivos y de presentación de información compleja y su aplicación a la resolución de problemas de diseño de interacción persona computadora..

  Esta competencia se obtuvo al realizar la adaptación de la visualización en Octave de cualquier modelo de la región IBI según determinadas profundidades e intervalos de tiempo del mismo.

Capítulo 4

Recursos utilizados

En este capítulo se enumerarán cada uno de los recursos utilizados para la realización de este proyecto.

4.1. Hardware

4.1.1. MacBook Pro

El proyecto lo comencé a desarrollar con este portátil del año 2015 que cuenta con un procesador i5 de 2.7Ghz, 8 GB 1867 MHz DDR3 y una gráfica integrada Intel Iris Graphics 6100 1536 MB. Al no necesitar en la gran parte del trabajo grandes capacidades de cómputo, constituía pues mi principal estación de trabajo.

4.1.2. Ordenador Sobremesa

Para la realización de la herramienta para la visualización de mapas he utilizado mi ordenador de sobremesa que cuenta con las siguientes especificaciones relevantes a la hora de realizar el cómputo:

- **Procesador**: Intel Core i7 6700K @4.6Ghz
- **Memoria Ram**: 32GB 2400MHz DDR4
- **Tarjeta Gráfica**: nVidia GTX 1080

Gracias a esta potencia los cálculos para la adaptación de la visualización eran más rápidos y podía trabajar y probar cambios con relativa soltura, exclusivamente se ha utilizado para este fin, el resto del trabajo se ha realizado sin mayor problema en el portátil.

4.2. Sistemas Operativos

4.2.1. MacOS High Sierra

Debo destacar el uso de este sistema operativo durante la realización del proyecto ya que posee ciertas diferencias con respecto a Windows 10 en el modo de instalación de los paquetes de Octave, pues no es capaz de hacer uso del compilador de Fortran debido a errores de compatibilidad al ser una versión del sistema operativo muy reciente. Es por ello por lo que me he decantado por la versión basada en C de netCDF para poder hacer uso del software Octave y sus paquetes externos necesarios.

4.2.2. Microsoft Windows 10

Como he comentado anteriormente, he utilizado el ordenador de sobremesa para poder realizar una parte del trabajo, en concreto la adaptación de la visualización de los modelos y para ello he contado con este sistema operativo por su facilidad de uso. Para mantener el esquema entre ambos sistemas operativos también he utilizado la versión basada en C de netCDF.

4.3. Software

4.3.1. MATLAB R2016b

MATLAB es un entorno de desarrollo de aplicaciones destinado a realizar cálculos matemáticos y visualizaciones gráficas. Se trata de una de las herramientas más conocidas y utilizadas para llevar a cabo funciones que hagan un uso intensivo de cálculos numéricos. Se trata además de una aplicación de pago.

4.3.2. GNU OCTAVE

OCTAVE también es un entorno de desarrollo de aplicaciones destinado a realizar cálculos matemáticos y visualizaciones gráficas, con la gran diferencia con respecto a MATLAB de que cuenta con una licencia de software libre, lo cual permite una distribución y modificación, así como ampliación del software de forma libre y sin exclusividades por parte del proveedor.

4.4. Librerías

Las librerías más utilizadas en OCTAVE son las siguientes:

- **netCDF**: Para la lectura, escritura y transformación de los archivos netCDF.
- **mapping**: Se utilizó para poder utilizar funciones de longitudes no existentes en OCTAVE de manera nativa.
- **Financial**: Utilizado para incluir métodos y procedimientos de intervalos de tiempo.

4.5. Otras herramientas utilizadas

4.5.1. BREW

Necesario para la instalación de Octave con el entorno de desarrollo gráfico y netCDF-CXX en el sistema operativo de macOS High Sierra.

4.5.2. netCDF-CXX

Paquete necesario para poder instalar y manejar la librería netCDF existente para Octave.

4.5.3. LATEX

Latex se ha utilizado para poder desarrollar la presente memoria, en concreto con el uso del editor TeX Maker.

4.5.4. StarUML

StarUML es una herramienta de desarrollo UML utilizado para el análisis y la creación de diagramas de actividad para el capítulo siguiente.

Capítulo 5

Análisis y Diseño

En este capítulo me centraré en el análisis y diseño que he tomado durante la realización del presente trabajo.

5.1. Esquema de trabajo inicial

Este proyecto se tomó con la idea de realizar diferentes herramientas de apoyo a la planificación de rutas óptimas, con dos elementos principales: los cargadores de mapas y los visualizadores.

En primer lugar, se han analizado las necesidades de los diferentes integrantes del grupo de investigación y tomado una lista de herramientas necesarias que se convierten en el principal esquema de trabajo del proyecto. A partir de ahí se ha trabajado en la construcción de unas funcionalidades básicas, en función del tiempo disponible y utilizando una metodología de desarrollo en espiral para poder seleccionar a posteriori funcionalidades adicionales para los diferentes prototipos, logrando así un crecimiento paulatino de la consistencia del proyecto.

Se llegaría como conclusión a dicha parte al siguiente esquema de trabajo:

- Inclusión de la librería netCDF en un software libre.
- Adaptación de funcionalidades básicas de dicha librería en relación a la línea de trabajo del equipo de investigación.
- Reanálisis e implementación de un nuevo sistema de indexado de modelos.
- Realización de la gestión de los proyectos del equipo de investigación en el software de Octave con un formato de texto siguiendo el esquema WYSIWYG.

- Realización de otra gestión de los proyectos paralela siguiendo el estándar XML, desarrollado nuevamente con el software Octave.
- Adaptación de la visualización de los modelos.
- Adaptación del núcleo del simulador.

Siguiendo el método de desarrollo en espiral anteriormente mencionado, antes del desarrollo de cada punto de la planificación se ha realizado un análisis previo de requerimientos y de implementación que se verá más detalladamente en el siguiente apartado. Cabe destacar que para la realización y adaptación de dichas herramientas ha sido necesario el estudio previo de las propias herramientas ya implementadas en el software de MATLAB por parte del equipo de investigación, así como del propio servicio CMEMS mencionado en anteriores capítulos.

5.2. Diseño

En la fase del diseño se han considerado varias opciones de implementación, todas ellas teniendo la premisa de que utilicen una licencia de software libre y tras barajar sobre las ventajas y desventajas de las mismas se ha llegado a la conclusión de utilizar Octave por tratarse de la herramienta más semejante a la implementación inicial del equipo de investigación, Matlab, logrando así maximizar la integración de las herramientas y evitando que el posterior uso por parte de los integrantes del equipo de investigación del IUSIANI conlleve una gran adaptación a las nuevas herramientas, mostrándose pues más familiarizados con el uso de las mismas.

Cabe destacar que para uno de los puntos de la gestión de los proyectos, una pequeña parte de demostración ha utilizado también el lenguaje de programación Python.

Es por ello por lo que tras este análisis de requisitos previo se ha conseguido que se pueda cumplir el objetivo principal de este trabajo de utilizar herramientas de software libre que no requieran el uso de licencias de pago. Y gracias al esquema de desarrollo en espiral se ha podido ir realizando las diferentes partes del esquema de trabajo de manera incremental, apoyándome con el tutor académico para ir avanzando en el desarrollo de las mismas.

5.3. Metodología

Como he comentado en la sección anterior, he seguido una metodología de desarrollo en espiral para realizar las diferentes herramientas en función

de las necesidades del equipo de investigación e incorporando a su vez más funcionalidades a medida que se desarrollaban.

Durante el transcurso del proyecto se han realizado varias reuniones donde se explicaba qué era lo que se debía realizar en cada herramienta y como poder abordar cada parte de las mismas.

- **Inclusión y adaptación de funcionalidades básicas netCDF en Octave**: Se debe adaptar cada una de las funcionalidades básicas de apertura y escritura de archivos netCDF, carga de datos del modelo correspondientes a la corriente marina, su localización y profundidad.

- **Reanálisis e implementación de un nuevo sistema de indexado de modelos**: Debe idearse un nuevo sistema de indexado de los modelos para la realización de futuras herramientas y a su vez facilitar la búsqueda de los modelos al equipo de investigación.

- **Realización de la gestión de los proyectos del equipo de investigación**: Debe realizarse una gestión de los proyectos existentes y futuros del equipo de investigación, ideando tanto cómo será su estructura de datos como la implementación de la misma usando Octave.

- **Adaptación de la visualización de los modelos**: Se debe adaptar la visualización de los modelos al software de Octave, permitiendo al investigador poder moverse tanto en la variable de profundidad del mapa del modelo como en los diferentes intervalos de tiempo del mismo, visualizando de una manera rápida y sencilla el modelo que desee.

- **Adaptación del núcleo del simulador**: Se debe adaptar el núcleo del simulador del equipo de investigación al software de Octave, tomando la premisa de que la precisión de los datos debe ser crucial para la exactitud de las pruebas que se realicen con el mismo.

5.4. Estructura de las herramientas

En esta sección analizaré la estructura global de las herramientas:

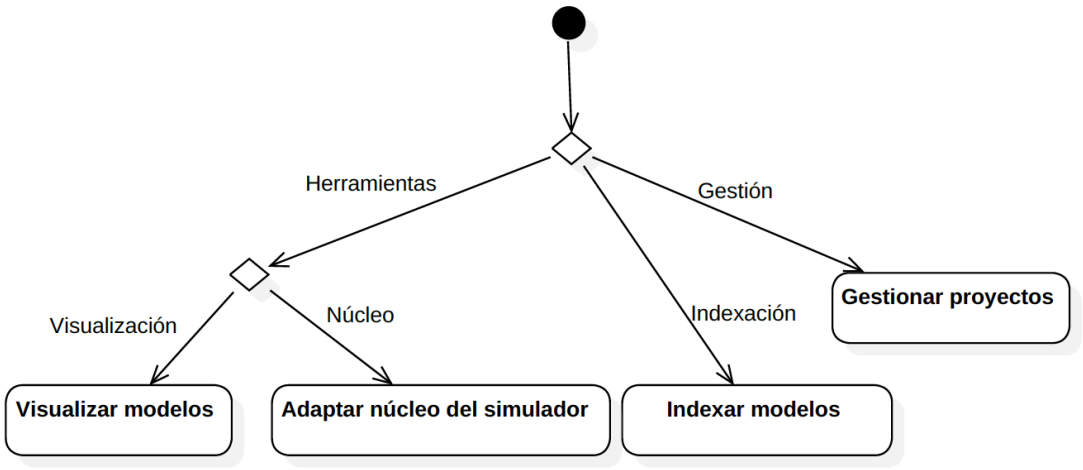

Figura 5.1: Diagrama de estructura global de herramientas

Como se ve en el diagrama anterior 5.1, el equipo de investigación dispone de una serie de herramientas para la visualización de modelos y utilización del núcleo del simulador, asimismo será capaz de buscar archivos de manera manual gracias al nuevo esquema de indexado que presentaré más adelante y podrá gestionar los proyectos que ya tiene así como incorporar proyectos futuros en una nueva estructura de directorios.

En siguiente lugar, especificaré cada uno de los diagramas de las diferentes funcionalidades del proyecto, empezando por el indexado de modelos:

Para el indexado de los modelos se pueden tomar dos aproximaciones en función de cuánta información se desee incorporar a la simple previsualización del nombre de los modelos, tomando como una primera aproximación la siguiente, ver figura 5.2:

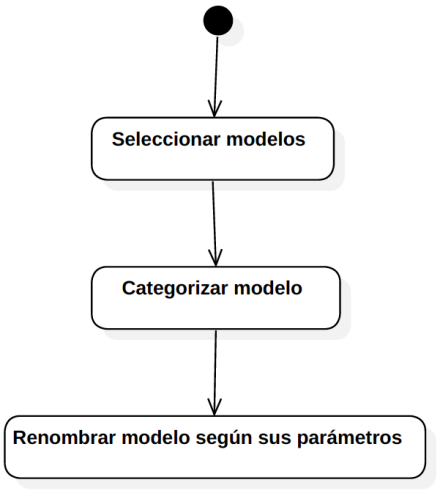

Figura 5.2: Primer diagrama indexado

Esta aproximación incrementa innecesariamente el número de elementos del nombre en la búsqueda de modelos. Aprovecharé la ventaja de la creación de una propia gestión de proyectos para establecer una segunda aproximación basada en el tiempo de creación del modelo y la propia fecha del mapa en la que el modelo se basa. En el capítulo de la implementación explicaré detalladamente cual es el proceso seguido para mejorar el funcionamiento ya desarrollado, ver figura 5.3

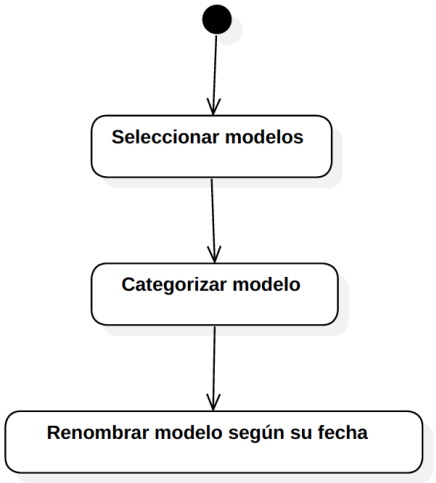

Figura 5.3: Segundo diagrama de indexado

Con respecto a la visualización de los modelos se ha seguido un diagrama de actividad algo diferente al propuesto por el equipo de investigación facilitando al usuario el manejo gracias a utilizar la propia ventana de comandos para poder adquirir de manera rápida y efectiva visualizaciones concretas de determinados modelos, ver figura 5.4

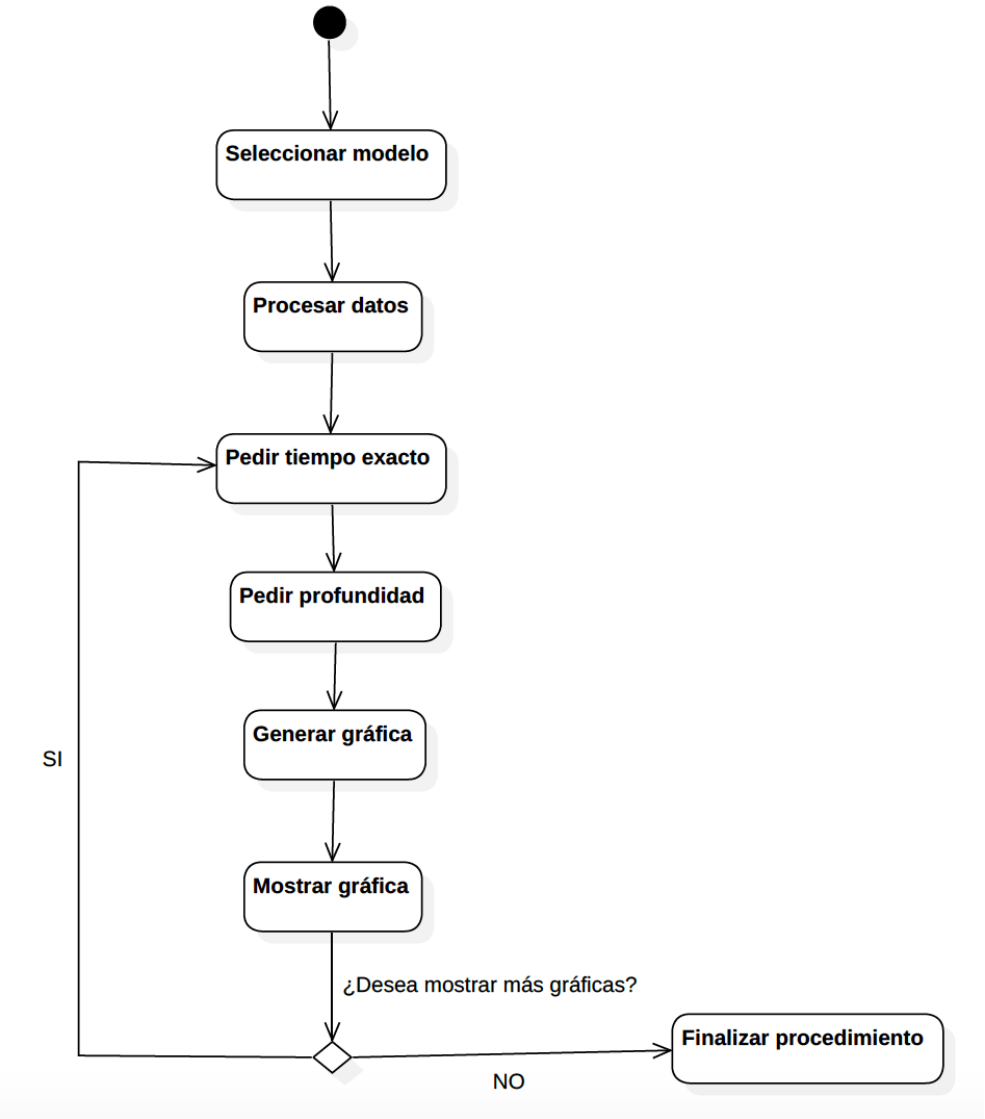

Figura 5.4: Diagrama de visualizado de modelos

Gracias a seguir este diagrama de actividad el usuario que utilice la herramienta podrá generar todas las gráficas que desee disminuyendo el tiempo

dedicado a la generación de las mismas por ser particular a los datos que introduzca el investigador.

En siguiente lugar, la estructura del núcleo del simulador sigue el siguiente diagrama de actividad, ver figura 5.5:

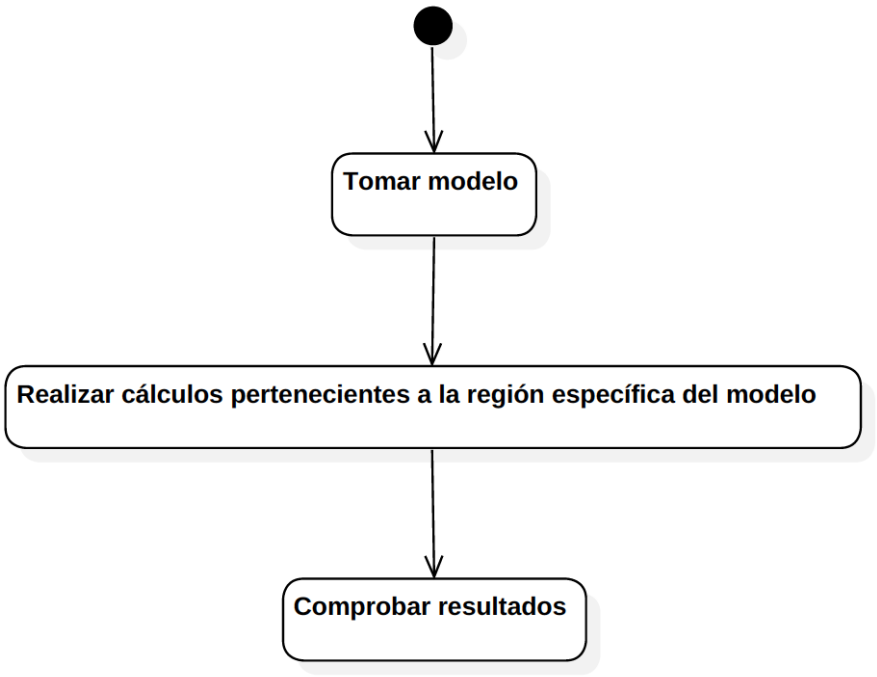

Figura 5.5: Diagrama del simulador

Se debe hacer hincapié en el último bloque de este diagrama, ya que la comprobación de los resultados es crucial para certificar que el núcleo del simulador devuelve los datos exactos a cualquier problema de simulación que se les presente a los investigadores.

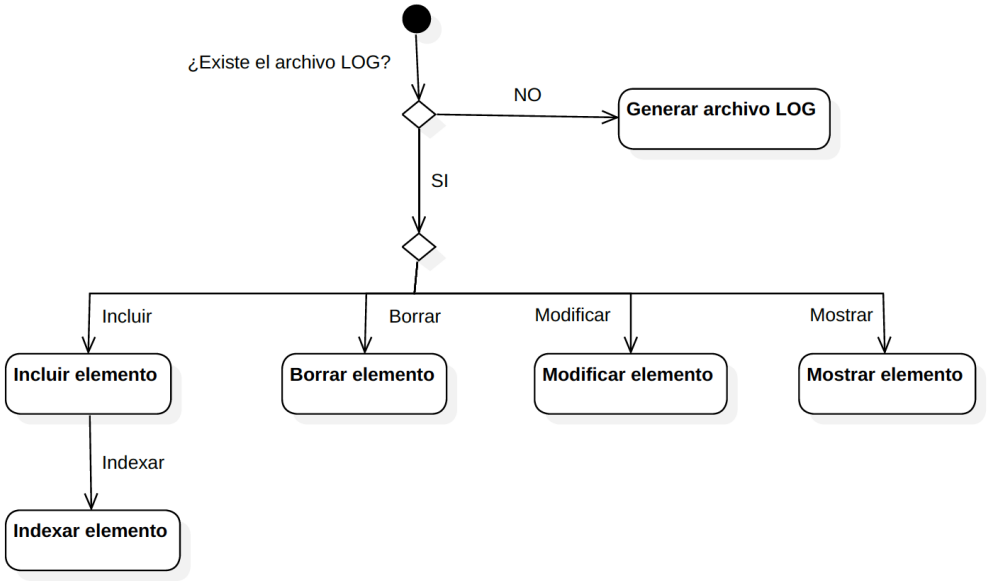

Figura 5.6: Diagrama del gestionado de proyectos

Con respecto a este último diagrama, ver figura 5.6, cabe destacar que se debe facilitar al equipo del IUSIANI la gestión de los proyectos, permitiendo seleccionar entre dos tipos de archivos de log que contendrán toda la información relativa tanto a los proyectos como a las misiones que contienen y los archivos que se utilicen en los mismos. Su implementación se detallará en el siguiente capítulo.

Capítulo 6

Implementación de las herramientas

En este capítulo detallaré cada una de las implementaciones y adaptaciones realizadas en el trabajo.

6.1. Descripción de las funciones utilizadas

En la siguiente sección hablaré sobre las funciones más importantes que se han utilizado a lo largo del trabajo que merecen especial atención tanto por su uso reiterado como por su importancia en el mismo.

6.1.1. Funciones relevantes netCDF utilizadas

- **netcdf_open** Esta función permite la apertura de archivos netCDF en un determinado modo especificado, los dos más importantes son **clobber**, que habilita poder sobrescribir el fichero y **noclobber**, el cual deshabilita esta opción. En el trabajo, por motivos de seguridad es de vital importancia marcar el segundo modo, pues no se deben modificar los archivos con los que el equipo de investigación se encuentre trabajando.

```
netcdf_open(archivo,modo);
```

- **netcdf_close** Esta función permite el cierre de un determinado archivo netCDF especificándole la id asociada al mismo.

```
netcdf_close(archivo.ncid);
```

- **netcdf_inqVarIDs** Esta rutina devuelve cada una de las identificaciones de las variables del archivo netCDF que se esté utilizando a partir de su identificador de archivo ncid.

  netcdf_inqVarIDs(archivo.ncid);

- **netcdf_inqVar** Este método permite obtener toda la información asociada a una variable netCDF, devolviendo el nombre, el tipo, el array de dimensiones y el número de atributos asociados a la misma.

  netcdf_inqVar(archivo.ncid,varID);

- **netcdf_getVar** Esta función obtiene los datos de la variable de un archivo netCDF. Por defecto si no se especifican más parámetros se obtendrán todos los elementos de dicha variable, en caso contrario se pueden determinar qué elementos son los que se desean adquirir. Para este trabajo se utilizará la opción por defecto. Existe una diferencia con MATLAB en el modo de uso de esta función, debido a que incorpora la opción de poder especificar el tipo de datos propio con el que se desee trabajar. En Octave es necesario realizar una conversión a posteriori de realizar la carga de los datos.

  netcdf_getVar(archivo.ncid,varID);

- **netcdf_getAtt** Este método devuelve el valor de un determinado atributo de una variable dentro de un determinado archivo netCDF.

  netcdf_getAtt(archivo.ncid,varID,nombreAtributo);

Todas estas funciones se encuentran presentes en el paquete externo de netCDF para Octave. Para poder hacer uso de las mismas es necesario instalar la interfaz netCDF como he comentado en anteriores capítulos, en concreto se ha utilizado la versión basada en C en su versión 4 utilizando la herramienta BREW en el sistema operativo de MacOS High Sierra. Seguidamente se ha instalado el paquete mediante el comando pkg install archivo-instalación-netcdf dentro del propio software de Octave.

Por otro lado, el uso de estas funciones se explicará en detalle en la sección dedicada a la primera herramienta, que incorpora y adapta las funcionalidades básicas para Octave y que han sido creadas por el equipo de investigación en el software de MATLAB.

6.1.2. Funciones relevantes Octave utilizadas

En esta sección se definirán aquellas funciones relevantes del trabajo que permiten trabajar con archivos, realizar tratamiento de cadenas de texto, comprimir, descomprimir, mostrar gráficas y transformar matrices numéricas:

- **gzip** Este método comprime los archivos que se pasen por parámetro en formato gzip en el directorio especificado. Esta función es útil al cerrar los archivos netCDF para comprimirlos y optimizar el espacio en disco.

  gzip(archivos,directorio);

- **gunzip** Esta función descomprime un archivo en formato gzip para su posterior uso. Su importancia radica en la apertura de los archivos netCDF y complementa a la función anteriormente definida.

  gunzip(archivo);

- **fileread** Este método obtiene la información de un determinado archivo del sistema en formato de cadena de caracteres.

  fileread(archivo);

- **fullfile** Esta función implementa de manera sencilla la unión de las cadenas de texto referidas a un directorio y al nombre de un archivo para formar una ruta completa.

  fullfile(directorio,archivo);

- **squeeze** Esta función permite eliminar las dimensiones singleton de la matriz x y devuelve el resultado. Una dimensión se considera singleton si posee tan solo longitud 1.

  squeeze(x);

- **permute** Este método permite permutar un vector según las posiciones dadas por el segundo parámetro.

  permute(x,permutacion);

- **meshgrid** Dados dos vectores x e y, genera matrices xx e yy, siendo las filas de la matriz xx copias de x; y las columnas de la matriz yy copias de y. En el caso de que no se especifique un vector y, tomará el vector x. Este método se puede aplicar también a un tercer vector z.

  meshgrid(x,y);

- **fix** Dada una matriz cualquiera x, trunca la parte fraccional de todos los valores y devuelve la porción entera de los mismos en una nueva matriz.

  fix(x);

- **quiver** Realiza un plot de vectores en dos dimensiones con flechas, en concreto lo hará en base a las componentes u,v de los campos obtenidos tras aplicar la función anterior de meshgrid. Si el mallado es uniforme, entonces x e y se pueden especificar como vectores y no como matrices. A continuación mostraré un ejemplo del resultado obtenido de la página oficial de octave [8].

  quiver(u,v);

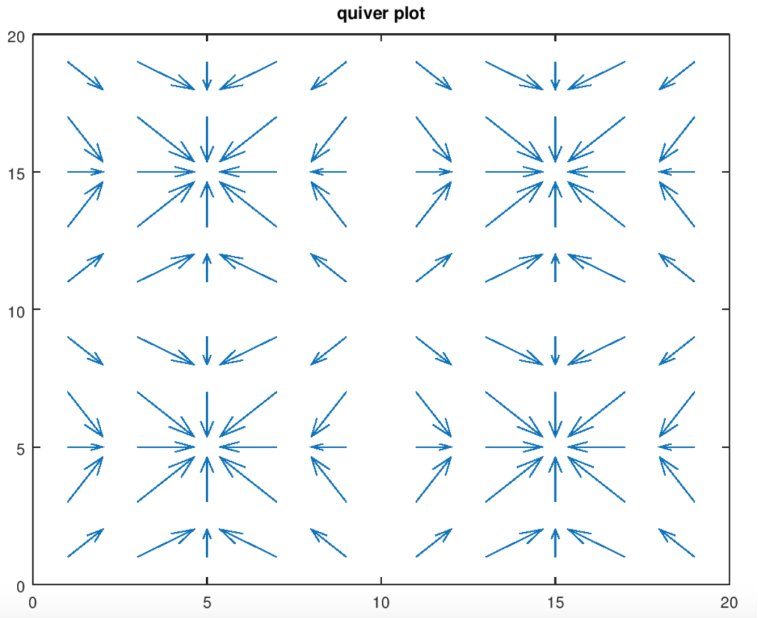

Figura 6.1: Ejemplo de un plot usando quiver

- **plot** Realiza una gráfica utilizando las componentes x e y, se le pueden aplicar a su vez estilos en la representación.

  plot(x,y);

- **line** Realiza una línea desde las posiciones x e y, dibujándola posteriormente en la gráfica que se encuentre activa en ese momento.

  line(x,y);

- **regexp** Busca patrones en una cadena de texto y devuelve las posiciones, así como las substrings encontradas.

  regexp(string,patron);

- **find** Busca elementos dentro de una matriz y devuelve los índices en forma de un vector donde encuentra las coincidencias.

  find(x);

- **strcmp** Devuelve 1 en caso de que dos cadenas de texto sean iguales o 0 en caso contrario

  strcmp(string1,string2);

- **substr** Devuelve una cadena de texto a partir de otra partiendo de una determinada posición hasta otra, esta otra posición se especifica mediante una longitud y no una segunda posición.

  substr(string, offset, longitud)

6.1.3. Funciones relevantes de otros paquetes externos utilizadas

Durante el transcurso del trabajo, se han utilizado librerías externas para poder solventar la carencia de algunas funciones de Octave que sí se encuentran presentes en Matlab, entre ellas se encuentra la propia librería netCDF anteriormente mencionada y dos librerías nuevas, mapping y financial.

Con respecto a la librería mapping se utilizó en concreto el siguiente método necesario para realizar las comprobaciones que comentaré más adelante.

- **deg2km** Este método obtiene en kilómetros la distancia en grados que se le especifique.

```
deg2km(grados)
```

En relación a la librería financial, se empleó la siguiente función:

- **hour** Esta función obtiene la hora en formato numérico de una determinada fecha. En concreto se ha utilizado al realizar gráficas en la herramienta destinada al visualizado de modelos.

```
hour(fecha)
```

6.2. Primera herramienta: adaptación de netCDF a Octave

En esta sección explicaré la implementación de la adaptación de netCDF y las funcionalidades básicas en el software de Octave.

6.2.1. Objetivos de la herramienta

En esta herramienta se desea adaptar funcionalidades básicas del equipo de investigación ya implementadas en el software de Matlab para Octave, así como incorporar todo el abanico de herramientas de ficheros netCDF en el mismo, al tratarse de un paquete no incorporado de manera nativa en dicho software computacional.

6.2.2. Explicación de la herramienta

Para poder realizar esta herramienta se ha tomado como conjunto de funcionalidades existentes las proporcionadas por el equipo de investigación del IUSIANI. Gracias a estos ficheros se puede establecer una adaptación tanto de las funciones elementales que se usan en Matlab de manera nativa y que deben ser adaptadas a las proporcionadas por el paquete netCDF, como de las funciones y procedimientos básicos para las operaciones con los diferentes modelos que se tomen del servicio CMEMS.

Las funcionalidades elementales que se adaptarán de Matlab a Octave, explicadas en la sección anterior son las siguientes:

- netcdf_open
- netcdf_close
- netcdf_inqVarIDs
- netcdf_inqVar
- netcdf_getVar
- netcdf_getAtt

Con respecto a las rutinas básicas que hacen uso las siguientes herramientas y que constituyen una base fundamental en el trabajo tenemos las siguientes:

- cesga_ibi_3d_load_depth
- cesga_ibi_load_current_component
- cesga_ibi_load_current
- cesga_ibi_load_location
- cesga_ibi_netCDF_close
- cesga_ibi_netCDF_open

6.2.3. Implementación de la herramienta

Con respecto a la implementación de la herramienta, se ha procedido de la siguiente manera:

En primer lugar se ha identificado a partir de las funciones proporcionadas y que han sido definidas en Matlab, aquellos métodos que deben ser adaptados a la nueva librería de Octave, así como las funciones básicas que los utilizan.

Para ello tomamos la función cesga_ibi_netCDF_open. En ella se hace uso del método de apertura de ficheros netCDF de la nueva librería netcdf_open, para el cual se le pasa por parámetro la ruta completa del archivo con extensión .nc, siendo el modo de apertura noclobber debido a que se desea no modificar los ficheros de modelos que se carguen en la herramienta por motivos de seguridad.

Listing 6.1: Apertura archivos netCDF

```
1 netCDF_file.ncid = netcdf_open([netCDF_file.file, ...
      '.nc'],'noclobber');
```

Se debe tener en cuenta que al realizar la apertura, es posible que el archivo se encuentre comprimido en formato gzip, por lo que previamente se debe descomprimir para su posterior apertura.

Listing 6.2: Descomprimir archivos netCDF

```
1 if gz
2     gunzip( [netCDF_file.file, '.nc.gz'] )
3 end
```

El funcionamiento general de esta función es el siguiente: Comprueba para el camino completo del archivo si éste existe y si se encuentra comprimido o

no, una vez hecho esto se crea una estructura en la que el campo file contiene la propia ruta. Tras ello, se abre el archivo almacenándolo en el campo ncid de la estructura.

La siguiente función que se desea adaptar es: cesga_ibi_netCDF_close. Ésta hace uso del método para cerrar archivos netCDF, por lo que nuevamente es necesario adaptar la función existente de cierre de Matlab por la nueva versión de Octave, a la cual se le pasa por parámetro el identificador del archivo, realizando su compresión y eliminando el fichero sin comprimir, ganando así espacio en disco a la hora de almacenar los archivos.

Listing 6.3: Cerrar archivos netCDF

```
1 netcdf_close( netCDF_file.ncid );
2 gzip( [netCDF_file.file, '.nc'] );
3 delete( [netCDF_file.file, '.nc'] );
```

El método cesga_ibi_load_location permite obtener la longitud y latitud de un determinado archivo netCDF. Éste método implementado en Matlab por el equipo, necesita nuevamente adaptación en el uso de la librería de Octave.

Se obtiene en primer lugar las variables a partir del identificador del archivo, se compara la cadena de caracteres 'longitude' con el campo del nombre de la variable obtenido con el método netcdf_inqVar, obteniendo así su índice.

Se realiza este mismo procedimiento para la variable 'latitude' y finalmente usando el método netcdf_getVar es posible obtener los datos de ambas variables, haciendo una posterior conversión al tipo de datos double:

Listing 6.4: Obtencion del indice de la longitud y carga de sus datos

```
1  lon_index = -1;
2    for varId = netcdf_inqVarIDs( netCDF_file.ncid )
3        if strcmp(netcdf_inqVar( netCDF_file.ncid, varId), ...
             'longitude') || ...
4                strcmp(netcdf_inqVar( netCDF_file.ncid, ...
                     varId), 'lon')
5            lon_index = varId;
6            break;
7        end
8    end
9
10 lon = double(netcdf_getVar( netCDF_file.ncid, lon_index));
```

Con respecto a la función cesga_ibi_load_current para la carga de las com-

ponentes de la corriente únicamente ha sido necesario modificar la subfunción cesga_ibi_load_current_component .

Para la cual se obtiene el índice de la componente de corriente con el mismo procedimiento anteriormente mencionado.

Tras esto la función obtiene el valor de relleno si existese, y si su valor entre 2 es mayor que el propio de cada uno de los valores de la matriz c, se descartarán usando el valor nulo.

Asimismo, en el caso de que exista un factor de escala, nuevamente se usará la función de Octave destinada a cargar el componente y multiplicarlo por toda la matriz c, siendo ésta la devuelta por la función.

Un ejemplo de uso del método netcdf_getVar y del método netcdf_getAtt dentro de la adaptación de la función es el siguiente:

Listing 6.5: Obtencion de variables y atributos

```
1 c = permute( ...
2       double(squeeze( netcdf_getVar( netCDF_file.ncid, ...
            component_index ) )), ...
3       [2, 1, 3] ...
4   );
5 try
6       scale = netcdf_getAtt(netCDF_file.ncid, ...
            component_index, 'scale_factor');
7       c  = c*scale;
8   catch error
9   end
```

Finalmente se tiene la función de carga de profundidad, esta función también requiere de cierta adaptación para su uso en Octave.

Para ello, se toma el índice de la variable 'depth' y obtenemos sus datos realizando la respectiva transformación al tipo de datos double.

Listing 6.6: Obtencion de profundidad

```
d_index = -1;
for varId = netcdf_inqVarIDs( netCDF_file.ncid )
    if strcmp(netcdf_inqVar( netCDF_file.ncid, varId), ...
        'depth')
        d_index = varId;
        break;
    end
end

z = double(netcdf_getVar( netCDF_file.ncid, d_index ));
```

Gracias a la realización de esta herramienta, se pueden sentar las bases para el desarrollo de las siguientes, pues todas hacen uso de estas funcionalidades básicas que permiten el control y manejo seguro de los archivos netCDF. Tras realizar un pequeño test, se puede comprobar que las herramientas se han adaptado a Octave con éxito, ver figura 6.2:

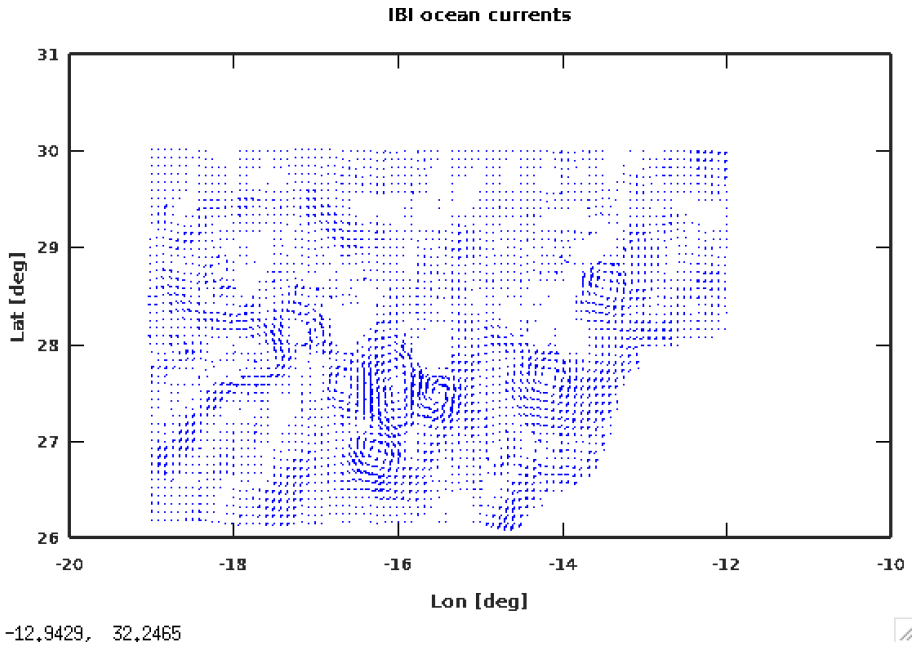

Figura 6.2: Primera herramienta

Se puede observar a simple vista que se ha cargado el modelo perteneciente a la región IBI, en concreto se ha mostrado la zona perteneciente al archipiélago canario, tomando como coordenadas la latitud y longitud, siendo los datos el mapa de corrientes con sus componentes u,v.

6.3. Segunda herramienta: indexado de modelos

Esta segunda herramienta se centrará en el indexado de los modelos netCDF.

6.3.1. Objetivos de la herramienta

Los principales objetivos en los que se centra el desarrollo de esta herramienta radican en rescribir el sistema de nombres de los modelos y mejorar su sistema de búsqueda para futuras consultas por parte de los investigadores.

6.3.2. Explicación de la herramienta

Para la realización de esta herramienta, se han tomado las funciones ya existentes de renombre de archivos para modelos 3D y 2D de cesga, así como el esquema 3D de hycom.

Tras analizar el modo de uso de las mismas, y comprobar que poseen un sistema de versiones incorporado, he decidido introducir una nueva versión de nombre para los modelos que permita un mejor indexado de los mismos en relación al esquema de la versión actual.

6.3.3. Implementación de la herramienta

La implementación de la herramienta es la siguiente:

Tras observar que aparecen campos repetidos en las dos versiones iniciales, en concreto el campo de la fecha real del modelo, se ha decidido tomar una nueva versión que no solo haga desaparecer el campo innecesario, sino que también mejore la ordenación de los archivos, efectuando así una mejora en la indexación de los modelos.

La versión actual utiliza el siguiente formato separado por barras:

- **pde**: Corresponde con la institución que crea el modelo, en este caso Puertos del Estado.
- **ibi35v1r1**: Correspondiente al tipo de versión que tiene el mapa regional sobre el que trata el modelo.
- **ibisr**: Región sobre la que el modelo posee datos.
- **01hav** o **01dav**: Pertenece al tipo de intervalo de tiempo que posee el modelo.

- **diaReal**: Día real del modelo.
- **diaReal**: Día real del modelo.
- **diaPetición**: Día en el que se realiza la petición del modelo.
- **FC**: Offset que indica el número de días de predicción del modelo.

Se puede observar a simple vista que esta versión es ineficiente tanto para la búsqueda de archivos concretos, como para su indexado, además de contener datos repetidos que hacen que esta búsqueda sea también costosa.

El modelo que he propuesto para las nuevas versiones de modelos es el siguiente:

- **diaReal**: Día real del modelo.
- **diaPetición**: Día en el que se realiza la petición del modelo.
- **pde**: Corresponde con la institución que crea el modelo, en este caso Puertos del Estado.
- **ibi35v1r1**: Correspondiente al tipo de versión que tiene el mapa regional sobre el que trata el modelo.
- **ibisr**: Región sobre la que el modelo posee datos.
- **01hav** o **01dav**: Pertenece al tipo de intervalo de tiempo que tiene el modelo.
- **FC**: Offset que indica el número de días de predicción del modelo.

Gracias a esta nueva versión, se permite la búsqueda de modelos a partir de la fecha real sobre la que se basa, al tratarse del dato más significativo del modelo, comprobando en su segundo valor la fecha desde la que se ha solicitado. Posteriormente añado todos y cada uno de los campos descriptivos del modelo sin necesidad de repetir cualquier campo.

De esta manera se facilita el indexado al seguir un esquema matricial de dos vectores, el vector asociado a la fecha real y el asociado a la fecha de petición, así el investigador verá reducido el tiempo de búsqueda, optimizando la estructura final de archivos netCDF del equipo.

Para realizarlo, he incorporado a las funciones cesga_ibi_2d_netCDF_filename , cesga_ibi_3d_netCDF_filename y hycom_3d_netCDF_filename una estructura switch en la que la versión 3 y la actual "0", sean la misma que estoy implementando, basándome en la estructura ya presente de dichos archivos.

Aquí se tiene un ejemplo de la estructura del archivo modificado cesga_ibi_2d_netCDF_filename :

Listing 6.7: Nueva version indexado

```
switch version
    case 1
        % Generate netCDF file name (old short form)
        str = fullfile( sprintf( ...
            'pde_ibi36v1r1_ibisr_01hav_%d%02d%02d_
            R%d%02d%02d_%cC0%d', ...
            dr, d, cast_charcode, cast_numcode ) );
    case 2
        % Generate netCDF file name (new long form)
        str = fullfile( sprintf( ...
            'pde_ibi36v1r1_ibisr_01hav_%d%02d%02d_
            %d%02d%02d_R%d%02d%02d_%cC0%d', ...
            dr, dr, d, cast_charcode, cast_numcode ) );
    case {0, 3}
        % Generate netCDF file name (new format)
        str = fullfile( sprintf( ...
            '%d%02d%02d_%d%02d%02d_pde_ibi36v1r1_
            ibisr_01hav_%cC0%d', ...
            dr, d, cast_charcode, cast_numcode ));
end
```

```
>> cesga_ibi_2d_netCDF_filename(0,[2018, 1, 8],3)
ans = 20180111_20180108_pde_ibi36v1r1_ibisr_01hav_FC04
>> cesga_ibi_3d_netCDF_filename(0,[2018, 5, 6],3)
ans = 20180509_20180506_pde_ibi36v1r1_ibisr_01dav_FC04
>> hycom_3d_netCDF_filename(0,[2018, 5, 6],3)
ans = 20180509_20180506_hycom
>>
```

Figura 6.3: Segunda herramienta

Se puede observar claramente el funcionamiento exitoso en la versión nueva del esquema de indexado de modelos netCDF de CMEMS en la figura anterior 6.3. En el caso de la función hycom se han obviado el resto de parámetros manteniendo el esquema de sus versiones anteriores.

6.4. Tercera herramienta: gestión de proyectos - texto

En esta tercera herramienta, implementaré desde cero la gestión de proyectos basado en un archivo de log en formato de texto siguiendo el patrón visual de WYSIWYG.

6.4.1. Objetivos de la herramienta

El objetivo de esta herramienta es proporcionar al investigador un control de la estructura de los datos de los proyectos y las misiones que contienen estos. Permitiendo realizar todo tipo de funciones de inclusión, borrado y modificación de los mismos, así como operaciones de visionado en el propio software de octave.

De esta manera se deberá conseguir implementar una librería bastante amplia y sencilla de utilizar, permitiendo además que el usuario final pueda modificar también de manera intuitiva el archivo de log directamente sin necesidad de utilizar Octave.

6.4.2. Explicación de la herramienta

En esta herramienta se creará un amplio abanico de métodos y procedimientos para la gestión de los proyectos, el listado de funciones es el siguiente:

- crearLog.m
- crearMision.m
- crearProyecto.m
- descripcionMision.m
- descripcionProyecto.m
- directoresMision.m
- directoresProyecto.m
- editarRutaArchivo.m
- eliminarArchivoMision.m
- eliminarArchivoProyecto.m

- eliminarMision.m
- eliminarMisionProyecto.m
- eliminarProyecto.m
- fechaFinalMision.m
- fechaFinalProyecto.m
- fechaInicioMision.m
- fechaInicioProyecto.m
- getArchivo.m
- getMision.m
- getProyecto.m
- incluirArchivoMision.m
- incluirArchivoProyecto.m
- incluirMisionProyecto.m
- participantesMision.m
- participantesProyecto.m
- publicacionesMision.m
- publicacionesProyecto.m
- renombrarMision.m
- renombrarProyecto.m
- rutaMision.m
- rutaProyecto.m

Para su implementación se han utilizado tanto las funciones básicas descritas en la primera herramienta, así como operaciones de tratamiento intensivo de cadenas de texto, detallados en el siguiente apartado.

6.4.3. Implementación de la herramienta

Con respecto a la implementación, en primer lugar se tomó cuál sería la estructura lógica del archivo que gestiona los proyectos, resultando de la siguiente manera para el caso de los proyectos:

- Proyecto
- Ruta del proyecto
- Descripción
- Fecha de Inicio
- Fecha de finalización
- Publicaciones realizadas
- Participantes
- Directores
- Misiones
- Fuentes

En el caso de las misiones, se sigue la siguiente estructura lógica similar a la anterior:

- Misión
- Ruta de la misión
- Descripción
- Fecha de Inicio
- Fecha de finalización
- Publicaciones realizadas
- Participantes
- Directores
- Fuentes

Para estructurar el trabajo de esta herramienta, he categorizado la librería en varios grupos. El primer grupo se corresponde con aquellas funciones que permiten la creación de elementos y los siguientes grupos corresponden con la edición tanto de archivos, como de proyectos y misiones.

Se tienen un total de 3 funciones nuevas para gestionar la creación de elementos:

- crearLog.m
- crearMision.m
- crearProyecto.m

Para la primera función, la cual permite crear y establecer un nuevo fichero de log, se le pasan dos parámetros referidos a la ruta donde se desea guardar el archivo y su nombre.

En primer lugar se realiza una comprobación de los tipos de los argumentos, para posteriormente crear el directorio si fuese necesario y guardar el archivo en el mismo.

Listing 6.8: Crear archivo log

```
1 function crearLog(rutaLog,nombreLog)
2   comprobarArgumentos(rutaLog,nombreLog);
3   archivo = strcat(nombreLog,'.txt');
4   mkdir(rutaLog);
5   salida = fullfile(rutaLog,archivo);
6   logId = fopen(salida,'w');
7   fclose(logId);
8 endfunction
```

Al realizar un tipo de programación modular, las funciones que defino se pueden usar en el resto de métodos. Un ejemplo es la función que comprueba los tipos de los argumentos, la cual se utiliza en casi todas las funciones de la librería.

Con respecto a las funciones que me permiten crear misiones y proyectos dentro del archivo de log, aprovecho el sistema matricial sobre el que Octave se sienta, ya que al estructurar el archivo por filas, puedo recorrer el mismo de una manera más optimizada.

Al crear una misión, se necesita especificar la ruta del archivo de log, la ruta donde se encuentra la misión y su nombre. Una vez definidas estas variables resulta trivial la incorporación del mismo en el archivo, pues compruebo si existe el log y añado las líneas asociadas a cada una de las variables de la misión:

Listing 6.9: Incluir mision

```
function crearMision(rutaLog,rutaMision,nombreMision)
  comprobarArgumentos(rutaLog,rutaMision,nombreMision);
  if exist(rutaLog) == 2
    logId = fopen(rutaLog,'a');
    fprintf(logId,'\n')
    fprintf(logId,'{MISION: \n');
    fprintf(logId,'     %s\n',nombreMision);
    fprintf(logId,'  RUTAMISION: \n');
    fprintf(logId,'     %s\n', rutaMision);
    fprintf(logId,'  DESCRIPCION: \n');
    fprintf(logId,'  FECHAINICIO: \n');
    fprintf(logId,'  FECHAFINAL: \n');
    fprintf(logId,'  PUBLICACIONES: \n');
    fprintf(logId,'  PARTICIPANTES: \n');
    fprintf(logId,'  DIRECTORES: \n');
    fprintf(logId,'  SOURCES: \n');
    fprintf(logId,'}\n');
    fclose(logId);
  else
    error('NO SE ENCUENTRA EL ARCHIVO LOG');
  endif
endfunction
```

Para incluir un proyecto se sigue el mismo procedimiento realizado al incluir la misión, con la salvedad de que se añadirán los campos asociados al proyecto.

El siguiente grupo a considerar es el relacionado con la edición de los archivos, el cual dispone de dos funciones, uno para editar la ruta del mismo y otro para obtener la información del mismo.

En la primera función, para editar la ruta de un archivo implemento varios métodos auxiliares, un primer método auxiliar destinado a la lectura del fichero log en forma de celdas mediante el uso de la función explicada al principio de este capítulo regexp().

En siguiente lugar, detecto todos y cada uno de los índices donde aparece el archivo gracias a la optimización explicada con anterioridad. Donde comparo para cada fila de las celdas si alguna contiene el nombre del archivo, y almaceno su índice gracias a la función find(). Con esto realizaría cambios para un mismo fichero en todo el archivo de log, lo cual evitaría un consumo de tiempo excesivo por parte del usuario ejecutando innecesariamente la misma función.

Listing 6.10: Detectar indices de archivo

```
function indiceArchivo = ...
    detectarArchivo(nombreArchivo,celdasLog)
  [X, indiceArchivo] = find(strcmp(celdasLog, ['    ' ...
      nombreArchivo]));
  if isempty(indiceArchivo) == 1
    error("ESTE ARCHIVO NO SE ENCUENTRA EN EL LOG");
  endif
endfunction
```

Finalmente reemplazo la ruta para cada una de las coincidencias encontradas, teniendo en cuenta que la posición de la ruta en la matriz de celdas del fichero log se encuentra justo en la posición i+1:

Listing 6.11: Reemplazar ruta archivo

```
function info = reemplazarRuta(infoAntigua,rutaNueva)
  indexRuta=strfind(infoAntigua{1},'Ruta: ')+5;
  info = [substr(infoAntigua{1},1,indexRuta) rutaNueva];
endfunction
```

Acabaré la implementación de la misma actualizando el fichero log en modo escritura.

En relación al método que devuelve la información relativa a cada archivo, se utilizan las funciones auxiliares anteriormente mencionadas para comprobar los tipos de los argumentos, obtener el fichero log en formato de celdas y los índices donde se encuentra el archivo, para finalmente leer la fila de la posición i+1 entera del primer elemento de la lista de índices donde encuentra una coincidencia en el archivo de log. Ya que éste contiene toda la información relativa al modelo del archivo, incluyendo sus variables.

A la hora de incluir un archivo en una misión o proyecto, estos campos se rellenarán en el archivo log realizando una lectura sobre los mismos, sin embargo, este procedimiento lo explicaré más adelante.

El siguiente grupo de funciones necesario es el que controla la gestión propia de los proyectos.

En primer lugar destaco la función que obtiene los datos de un proyecto y los devuelve en forma de una estructura nativa de Octave completa, pues es útil a la hora de visualizar en el propio software cada uno de los elementos que componen un proyecto y poder realizar operaciones a posteriori sobre los mismos.

Para ello, se realizan las comprobaciones de los tipos de los argumentos y se obtiene tanto el archivo log como el índice del nombre del proyecto.

Una vez hecho esto, se sigue un procedimiento similar por cada uno de los

campos que forman el proyecto, excepto para el que obtiene la información de los archivos, ya que difiere en una parte.

Para cualquier campo se busca el identificador del mismo y se obtienen todos sus índices en el archivo de log, por ejemplo para el caso de la descripción del proyecto se buscaría "DESCRIPCION". Tras ello, sólo nos quedaremos con aquel índice justamente superior al propio del proyecto, pues el resto nos resultan irrelevantes.

Se realizará una pequeña comprobación para dicho campo, ya que puede no contener datos si así se desea, y finalmente se devuelve el dato asociado.

Listing 6.12: Obtener datos de un campo del proyecto

```
function descripcion = getDescripcion(M,celdasLog)
  [I, J] = find(strcmp(celdasLog, '  DESCRIPCION: '));
  for indice=J
    if (indice > M)
      indiceDescripcion = indice;
      break
    endif
  endfor
  if strcmp(celdasLog{indiceDescripcion+1},
  ['  FECHAINICIO: ']) == 0
    descripcion = [substr(celdasLog{indiceDescripcion+1},5,
    length(celdasLog{indiceDescripcion+1})-4)];
  else
    descripcion = [];
  endif
endfunction
```

Para el resto de campos se sigue una metodología similar a la anterior expuesta. En el caso de la información de los archivos, se hace necesario realizar búsquedas dentro de cadenas de caracteres, debido a que se encuentra en la misma línea, esto es así ya que si se decidiese utilizar el sistema de dato por fila, el archivo log se haría ininteligible, por lo que rompería con uno de los objetivos principales que busca la propia edición directa del usuario sobre el fichero de log.

Otro método importante es el que me permite extraer información relevante sobre los archivos para poder plasmarlo en el fichero de log.

Éste realiza nuevamente las comprobaciones iniciales y obtiene con los métodos expuestos anteriormente los índices tanto del proyecto donde se desea incluir el archivo, como del campo destinado a ello, es este caso el campo "SOURCES".

Tras esto, el procedimiento de actuación es trivial teniendo en cuenta cada uno de los métodos anteriormente mencionados, con la salvedad de que para

extraer la información del fichero cargo la librería de Octave netCDF, realizo la apertura del archivo según la ruta y su nombre con la función básica desarrollada en la primera herramienta: cesga_ibi_3d_netCDF_open_octave y obtengo la información usando una nueva función de la librería ncinfo(archivo).

Esta función genera una estructura de datos con toda la información relativa al fichero. Como deseo extraer únicamente su información relevante, en este caso las variables que contiene, obtengo cada uno de los nombres de las mismas a partir del campo Name de la estructura.

Finalmente se recorrerían las celdas e incluirían en un vector para posteriormente introducirlo en el archivo. Este método concluiría a su vez con el uso de la función básica que me permite cerrar y comprimir el archivo luego de su uso.

Listing 6.13: Obtener datos de archivo

```
1  function variables = getInfoFile(nombre,ruta)
2    pkg load netcdf;
3    archivo = cesga_ibi_3d_netCDF_open_octave([ruta '/' ...
         nombre]);
4    info_archivo = ncinfo([ruta '/' nombre '.nc']);
5    variables_cell = {info_archivo.Variables.Name};
6
7    variables = [];
8
9    for i=variables_cell
10     variables = [variables i{1} ' '];
11   endfor
12
13   cesga_ibi_3d_netCDF_close_octave(archivo);
14 endfunction
```

En alusión al resto de funciones tanto para la inclusión y borrado de elementos, así como de modificación de los mismos se utiliza el mismo esquema de desarrollo, sin embargo, en la siguiente sección en la que realizo la misma herramienta para la gestión de proyectos siguiendo el estándar XML, se difiere en diversas partes a la hora de la implementación.

En el apéndice del documento se mostrará un manual de usuario detallado sobre el uso de cada método de ambas librerías.

Un ejemplo de archivo de log usando esta herramienta es el siguiente, ver figura 6.4

```
{MISION:
    mision2
  RUTAMISION:
    /rutam2
  DESCRIPCION:
  FECHAINICIO:
  FECHAFINAL:
  PUBLICACIONES:
  PARTICIPANTES:
  DIRECTORES:
  SOURCES:
    mision2_20170724_R20170724_FC01.nc.gz
      Region: IBISR Tipo: CESGA Variables: zos mlotst time so thetao longitude bottomT
latitude uo depth vo Ruta: /prueba/mision21
    mision2_FC01.nc.gz
      Region: IBISR Tipo: CESGA Variables: zos mlotst time so thetao longitude bottomT
latitude uo depth vo Ruta: /prueba/mision22
    mision2_3333_FC01.nc.gz
      Region: IBISR Tipo: CESGA Variables: zos mlotst time so thetao longitude bottomT
latitude uo depth vo Ruta: /prueba/mision333333
}

{PROYECTO:
    pro2
  RUTAPROYECTO:
    /p2
  DESCRIPCION:
    Esto es un campo descriptivo del proyecto.
  FECHAINICIO:
    04/06/1996
  FECHAFINAL:
    08/09/2000
  PUBLICACIONES:
    Pub1
    Pub2
  PARTICIPANTES:
    Par1
  DIRECTORES:
    Uno
  MISIONES:
     mision1
     mision2
  SOURCES:
    pde_ibi36v1r1_ibisr_01dav_20170724_20170724_R20170724_FC01.nc.gz
      Region: IBISR Tipo: CESGA Variables: zos mlotst time so thetao longitude bottomT
latitude uo depth vo Ruta: /prueba/Ruta3
}
```

Figura 6.4: Tercera herramienta

6.5. Cuarta herramienta: gestión de proyectos - XML

En esta cuarta herramienta, implementaré a partir de la desarrollada anteriormente, una gestión de proyectos basado en un archivo log que sigue el estándar de almacenamiento de datos XML.

6.5.1. Objetivos de la herramienta

El objetivo de la herramienta consistirá nuevamente en proporcionar al investigador un control de la escritura de los metadatos de los proyectos y misiones de los mismos. Incluyendo las funciones ya explicadas en la anterior herramienta y permitiendo además su lectura en diversas aplicaciones y no solo en Octave.

Se mostrará un ejemplo de lectura basado en Python.

6.5.2. Explicación de la herramienta

Para la realización de esta herramienta parto del concepto anteriormente propuesto en la herramienta 3, siguiendo un esquema de librería tal como sigue, manteniendo así la facilidad de uso para el investigador, pues puede llamar exactamente a las mismas funciones y de la misma manera que en la herramienta anterior, fomentando de esta manera una mejor integración global de las herramientas con diversos sistemas de log:

- crearLog.m
- crearMision.m
- crearProyecto.m
- descripcionMision.m
- descripcionProyecto.m
- directoresMision.m
- directoresProyecto.m
- editarRutaArchivo.m

- eliminarArchivoMision.m
- eliminarArchivoProyecto.m
- eliminarMision.m
- eliminarMisionProyecto.m
- eliminarProyecto.m
- fechaFinalMision.m
- fechaFinalProyecto.m
- fechaInicioMision.m
- fechaInicioProyecto.m
- incluirArchivoMision.m
- incluirArchivoProyecto.m
- incluirMisionProyecto.m
- participantesMision.m
- participantesProyecto.m
- publicacionesMision.m
- publicacionesProyecto.m
- renombrarMision.m
- renombrarProyecto.m
- rutaMision.m
- rutaProyecto.m

Para su implementación, nuevamente se han utilizado las funciones básicas de tratamiento de archivos netCDF, así como las destinadas al tratamiento de cadenas de caracteres, haciendo especial hincapié en mantener una correcta estructura del formato XML.

6.5.3. Implementación de la herramienta

Con respecto a su implementación, se ha seguido el esquema de campos propuesto con anterioridad, siguiendo el formato XML, para lo cual se han realizado validaciones utilizando la página oficial de w3schools [9]. Obteniendo de esta manera para el caso de los proyectos el siguiente modelo, ver figura 6.5

```
<PROYECTO>
  <TITULOPROYECTO></TITULOPROYECTO>
  <RUTAPROYECTO></RUTAPROYECTO>
  <DESCRIPCION></DESCRIPCION>
  <FECHAINICIO></FECHAINICIO>
  <FECHAFINAL></FECHAFINAL>
  <PUBLICACIONES></PUBLICACIONES>
  <PARTICIPANTES></PARTICIPANTES>
  <DIRECTORES><DIRECTOR></DIRECTOR><DIRECTOR></DIRECTOR></DIRECTORES>
  <MISIONES><MISIONPROYECTO></MISIONPROYECTO><MISIONPROYECTO></MISIONPROYECTO></MISIONES>
  <SOURCES></SOURCES>
</PROYECTO>
</LOG>
```

Figura 6.5: Esquema proyecto

Siendo éste, el caso dedicado a las misiones, ver figura 6.6

```
<MISION>
  <TITULOMISION></TITULOMISION>
  <RUTAMISION></RUTAMISION>
  <DESCRIPCION></DESCRIPCION>
  <FECHAINICIO></FECHAINICIO>
  <FECHAFINAL></FECHAFINAL>
  <PUBLICACIONES></PUBLICACIONES>
  <PARTICIPANTES></PARTICIPANTES>
  <DIRECTORES></DIRECTORES>
  <SOURCES></SOURCES>
</MISION>
```

Figura 6.6: Esquema mision

Nuevamente, se tienen 3 grandes grupos dentro de la librería de gestión de proyectos. Un primer grupo que permite la creación de los elementos y dos grupos destinados a las operaciones tanto de proyectos como de misiones.

Con respecto al primer grupo se tienen 3 funciones:

- crearLog.m
- crearMision.m

- crearProyecto.m

Siendo el primer método el usado para crear un archivo de log a partir de una dirección y un nombre. Este método difiere únicamente con respecto al proporcionado por la herramienta 3 en que deberá incluir una cabecera XML nada más realizar la creación del mismo, por lo que no debe devolverse un archivo vacío.

Listing 6.14: Crear archivo log XML

```
function crearLog(rutaLog,nombreLog)
  comprobarArgumentos(rutaLog,nombreLog);
  archivo = strcat(nombreLog,'.xml');
  mkdir(rutaLog);
  salida = fullfile(rutaLog,archivo);
  logId = fopen(salida,'w');
  fprintf(logId,'<?xml version="1.0" ...
     encoding="UTF-8"?>\n<LOG>\n</LOG>')
  fclose(logId);
endfunction
```

Cabe destacar que la mayoría de funciones auxiliares heredan de la tomada en el apartado anterior, por lo que no es necesario implementarlas de nuevo gracias al esquema modular de desarrollo.

Por otro lado, la implementación difiere en ciertos aspectos, ya que cada línea del archivo no corresponde al esquema: (Posición i = Nombre del campo, Posición i+1 = Valor del campo), sino que se empareja en una misma línea, lo que provoca una operación extra de tratamiento de cadenas de caracteres para encontrar los valores y separarlos del nombre del campo asociado.

En el caso de la creación de misiones, hay que tener en cuenta que se deben introducir a priori de la etiqueta de cierre de log, por lo que se buscará la posición de dicho elemento en el archivo, el cuál sigue manteniendo una estructura matricial de celdas, y se insertará tanto la misión con los valores iniciales de nombre y ruta, como la etiqueta de cierre del log.

Listing 6.15: Crear mision XML

```
function crearMision(rutaLog,rutaMision,nombreMision)
  comprobarArgumentos(rutaLog,rutaMision,nombreMision);
  celdasLog = obtenerLog(rutaLog);
  [X, W] = find(strcmp(celdasLog, '</LOG>'));
  celdasLog = {celdasLog{1:W(end)-1},[''],['<MISION>'],['  ...
     <TITULOMISION>',nombreMision,'</TITULOMISION>'],...
```

```
6               [' ...
                   <RUTAMISION>',rutaMision,'</RUTAMISION>'],[' ...
                    <DESCRIPCION></DESCRIPCION>'],...
7               [' <FECHAINICIO></FECHAINICIO>'],[' ...
                   <FECHAFINAL></FECHAFINAL>'],...
8               [' <PUBLICACIONES></PUBLICACIONES>'],[' ...
                   <PARTICIPANTES></PARTICIPANTES>'],...
9               [' <DIRECTORES></DIRECTORES>'],[' ...
                   <SOURCES></SOURCES>'],['</MISION>'],['</LOG>']};
10
11   actualizarLog(rutaLog,celdasLog);
12
13 endfunction
```

Para la creación de proyectos se sigue exactamente el mismo esquema de implementación.

Con respecto a los dos grupos restantes dentro de la nueva librería, detallaré el método que me permite establecer directores dentro de un proyecto, en concreto la subrutina que realiza dicha modificación.

Para realizar la modificación pertinente se parten de varios parámetros esenciales: la matriz de celdas del archivo log, una bandera que indica si se desean añadir directores sobre los ya existentes en el proyecto o directamente reemplazarlos, así como los índices donde se encuentra el campo de directores y el del proyecto. La obtención de estos dos últimos parámetros se realiza utilizando los métodos auxiliares ya definidos en la herramienta anterior.

En primer lugar se comprueba si se desea reemplazar o no los directores. Si es verdadero este valor, se comienza la generación de la fila, introduciendo la etiqueta general "DIRECTORES", recorriendo a posteriori cada uno de los directores e introduciéndolos con sus respectivas etiquetas, para finalmente tomar de la matriz de celdas del log todas las filas anteriores a esta, introducir la nueva y nuevamente las filas siguientes a la misma.

Listing 6.16: Establecer directores con reemplazamiento

```
1   if reemplazar == true
2     subindice=0;
3     celdaDirectores = ['  <DIRECTORES>'];
4     for director=directores
5       celdaDirectores = [celdaDirectores,'<DIRECTOR>',
6       directores{subindice+1},'</DIRECTOR>'];
7       subindice=subindice+1;
8     endfor
9     celdaDirectores = [celdaDirectores,'</DIRECTORES>'];
10    celdasLog = ...
          {celdasLog{1:indiceDirectores-1},celdaDirectores,
```

```
11      celdasLog{indiceDirectores+1:end}};
```

Para el caso de mantener los directores existentes, simplemente habría que tomar la propia celda de los directores según su índice obtenido mediante los métodos ya comentados en la herramienta anterior y almacenarlo en un vector.

A este vector se le aplicará una búsqueda de la substring del cierre de la etiqueta principal "DIRECTORES", obteniendo su índice y posteriormente gracias al mismo eliminarlo para introducir los nuevos directores.

Esta inclusión se hará a partir de la última posición del vector, incorporando justo al final la etiqueta de cierre anteriormente eliminada.

Finalmente se actualiza la matriz de celdas con los nuevos cambios.

Listing 6.17: Establecer directores sin reemplazamiento

```
subindice = 0;
    celdaDirectores = [celdasLog{indiceDirectores}];
    indiceFinal = strfind(celdaDirectores,'</DIRECTORES>');
    celdaDirectores = celdaDirectores(1:indiceFinal-1);
    for director=directores
      celdaDirectores = ...
          [celdaDirectores,'<DIRECTOR>',directores{subindice+1},
      '</DIRECTOR>'];
      subindice=subindice+1;
    endfor
    celdaDirectores = [celdaDirectores,'</DIRECTORES>'];
    celdasLog = ...
        {celdasLog{1:indiceDirectores-1},celdaDirectores,
    celdasLog{indiceDirectores+1:end}};
```

El resto de funciones implementadas en esta librería siguen tanto el esquema mostrado en la herramienta anterior por las funciones auxiliares como las pequeñas diferencias introducidas en las contempladas en esta cuarta herramienta. He mostrado el ejemplo de la introducción de directores pues incluye todos los casos posibles, siendo estos la introducción, modificación y eliminación de elementos.

La eliminación de elementos vuelve a tratarse de una modificación de la línea buscada, pues sólo establece los valores por defecto (las etiquetas generales).

Finalmente, para la lectura de los datos del fichero de log ya no se hace necesario que se haga usando Octave, cualquier herramienta podría realizarlo. En este caso mostraré un ejemplo de lectura de un archivo de log basado en Python.

Listing 6.18: Test XML Python

```
1 import xml.dom.minidom
2 archivoLog = ...
      xml.dom.minidom.parse('/Users/prueba/Desktop/log1.xml')
3 xmlEntero = archivoLog.toprettyxml()
4 print(xmlEntero)
```

El resultado siguiente, ver figura 6.7, muestra un extracto de la ejecución del código anterior. Esta herramienta se ha implementado con éxito, superando además las pruebas de validación de los formatos XML de w3schools [9].

```
<FECHAINICIO/>


<FECHAFINAL/>


<PUBLICACIONES>
    <PUBLICACION>Pub1</PUBLICACION>
</PUBLICACIONES>


<PARTICIPANTES>
    <PARTICIPANTE>Nombre1</PARTICIPANTE>
    <PARTICIPANTE>Nombre2</PARTICIPANTE>
</PARTICIPANTES>


<DIRECTORES>
    <DIRECTOR>Nombre4</DIRECTOR>
    <DIRECTOR>Nombre3</DIRECTOR>
    <DIRECTOR>Nombre1</DIRECTOR>
</DIRECTORES>


<SOURCES>
    <SOURCE>
        <NOMBRESOURCE>pde_ibi36v1r1_ibisr_01hav_20170724_20170724_R20170724_FC01.nc.gz</NOMBRESOURCE>
        <REGION>IBISR</REGION>
        <TIPO>CESGA</TIPO>
        <VARIABLES>
            <VAR>zos</VAR>
            <VAR>mlotst</VAR>
            <VAR>time</VAR>
            <VAR>so</VAR>
            <VAR>thetao</VAR>
            <VAR>longitude</VAR>
            <VAR>bottomT</VAR>
            <VAR>latitude</VAR>
            <VAR>uo</VAR>
            <VAR>depth</VAR>
            <VAR>vo</VAR>
        </VARIABLES>
```

Figura 6.7: Extracto XML python

6.6. Quinta herramienta: visualizado de modelos

Esta quinta herramienta corresponde a una adaptación de la ya existente y utilizada por el equipo de investigación para la visualización de modelos de corrientes.

6.6.1. Objetivos de la herramienta

Los objetivos principales de la herramienta consisten en adaptar el formato de visualización de modelos netCDF utilizado en MATLAB y realizarlo de una manera en la que se pueda comprobar tanto la profundidad como la hora exacta del intervalo de muestra.

6.6.2. Explicación de la herramienta

Para realizar esta herramienta se ha partido de la ya utilizada por el equipo de investigación del IUSIANI, comprendiendo la estructura de modelos que utiliza, así como el bloque dedicado a la parte gráfica y de visualización de la misma.

6.6.3. Implementación de la herramienta

Con respecto a la implementación, se deben tener en cuenta varios factores. El primero radica en la obtención propia de los modelos, pues se parte de una estructura de directorios que divide los datos relativos a las regiones y los datos propios de los modelos, distribuidos según sus fechas.

La estructura con la que se ha trabajado, así como los modelos son los siguientes, ver figura 6.8 :

Figura 6.8: Estructura de directorios

Por otro lado, el factor de la adaptación de nuevas funciones básicas es importante de cara tanto a esta herramienta como para la siguiente del núcleo del simulador.

Finalmente, la visualización gráfica en Octave es crucial ya que existen diferencias con respecto a Matlab a la hora de poder mostrar gráficamente los resultados y previsualizaciones de los modelos.

En relación a la adaptación de las funciones básicas, tenemos las siguientes:

- ComputeLonLatIndex
- ComputeUVZH_IBI_D2
- ComputeUVZH_IBI_H
- ComputeUVZH_IBI
- GetAreaInfo
- IBIZCurrH4
- LoadUV_2D_IBI
- LoadUV_3D_IBI
- LoadUVZH
- ShowCurr_4D

ShowCurr correspondería con la función principal que sirve para mostrar las previsualizaciones de los modelos; GetAreaInfo permitiría cargar datos referentes a la región sobre la que se deben realizar los cálculos; Las funciones de carga tomarán del modelo de la estructura de directorios las variables de corrientes para finalmente realizar los cálculos mediante las funciones de cómputo.

He decidido no incluir en la lista la rutina mostrarGrafico, pues ésta no se encuentra separada en el código del equipo de investigación, sino que se encuentra incluido dentro del propio método ShowCurr. Esta separación se debe a que si sigo un esquema de desarrollo modular, no tiene sentido realizar cálculos extra en una función que únicamente sirve para visualizar los modelos, permitiendo así poder modificar en el futuro la función de muestra a conveniencia sin necesidad de alterar el resto del código por motivos de seguridad.

Con respecto a las funciones básicas de carga de corrientes y su cálculo, no se ha realizado ningún tipo de modificación en relación al modelo propuesto por el IUSIANI, sin embargo, al hacer uso dichas funciones de las básicas desarrolladas en la primera herramienta, se puede afirmar que de forma transparente se ha realizado la adaptación a Octave, pues ésta es la que se encarga realmente de la toma de la información de los archivos netCDF.

En relación a la rutina mostrarGrafico, sí se han realizado diversos cambios, pues para facilitar y optimizar la visualización de gráficas empleo un método de encuestas (Polling) en lugar de por interrupciones, permitiendo al usuario tras cada muestra elegir tanto la hora exacta que desee ver como la profundidad.

Uno de los principales motivos para realizarlo de esta manera radica en la facilidad de implementación, pues resulta trivial su uso. Por otro lado, en el caso de tener un modelo con un intervalo de tiempo muy grande, el método propuesto te situaría al inicio del vector de tiempo, y no realmente donde el investigador quiere posicionarse, por lo que brindando esta posibilidad de elegir día y hora exactos se resuelve el problema, ahorrando mucho tiempo de búsqueda al investigador, pues se necesita una herramienta eficiente. Esto mismo se aplica para la segunda variable de muestra, la profundidad.

Explicaré ahora el fragmento de código desarrollado por el equipo de investigación en el IUSIANI que realiza las gráficas individuales: en el código se situarán las coordenadas de la gráfica en función de la latitud y longitud de la región sobre la que se trabaja, se tomaran las variables de corrientes en función de que sean en 3D con tiempo, 2D con tiempo, 2D con profundidad o 3D con tiempo y profundidad.

Se realizará luego el método para obtener el mallado a dibujar y se dibujarán con el método quiver comentado anteriormente, así como el título y leyendas de la gráfica.

Por lo tanto, este fragmento de código ya desarrollado e implementado en Matlab, únicamente se ha extraído para permitir modularizarlo e introducido la función hora() no existente en Octave.

Esta separación permite su incorporación en el método Polling que he realizado para gestionar las muestras de las mismas:

Listing 6.19: Metodo Polling de visualizado

```
1 while 1,
2   disp("+++INTRODUCCION DE DATOS:")
3   dias=numel(times)/24;
4   if dias > 1
5     d=input(["Introduce un dia del intervalo entre 1 y
6     " num2str(dias) ": "]);
```

```
7     else
8       d=1;
9     end
10    hora=input("Introduce una hora entre 0 y 23: ")+1;
11    iH=(d-1)*24+hora;
12    zI=input(["Introduce una profundidad entre 1 y
13    " num2str(nDepths) ": "]);
14    mostrarGrafico(llLim,mUVZH,aInfo,step,times,
15    depths,lats,lons,MAX_CURR,scale,color,refLL,
16    thr,coast,model,iH,zI);
17    disp("+++Grafico mostrado correctamente+++")
18    next=input(" Mostrar otro grafico [s,n]: ","s");
19    if next == 'n'
20      break;
21    endif
22  endwhile
```

En él se ve claramente como solicito los datos que se desean ver del modelo, y llamo a la función que muestra el gráfico especificado, permitiendo tanto terminar con la función como mostrar otro gráfico diferente de corrientes.

Aquí se muestra un ejemplo de uso, ver figura 6.9:

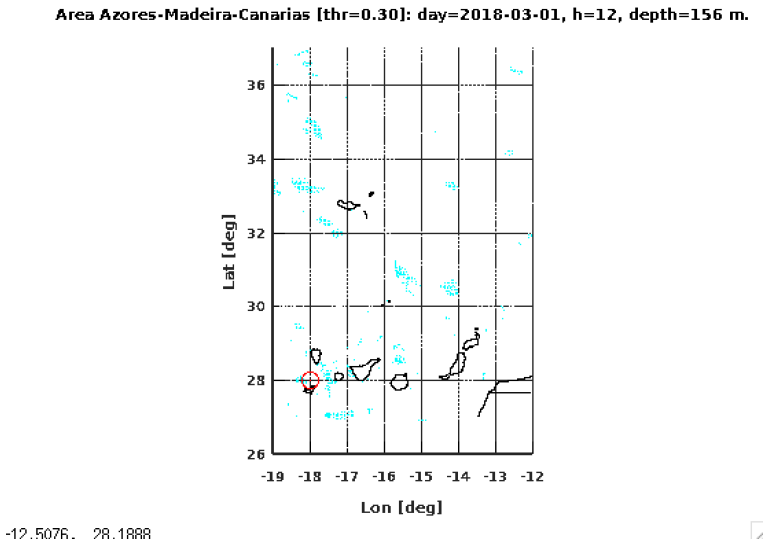

Figura 6.9: Ejemplo visualizado 1

Aquí otro ejemplo de uso con una hora y profundidad diferente al anterior sin tener que volver a calcular todas las matrices de valores, ver figura 6.10

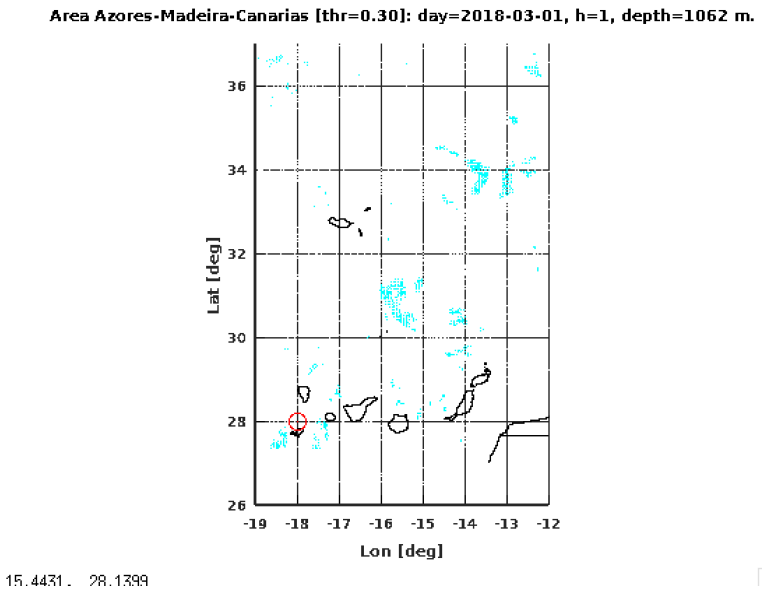

Figura 6.10: Ejemplo visualizado 2

6.7. Sexta herramienta: núcleo del simulador

En esta sección se implementará la sexta y última herramienta del trabajo, el núcleo del simulador.

6.7.1. Objetivos de la herramienta

El núcleo del simulador es una de las partes que tienen mayor relevancia dentro del equipo de investigación, pues sus cálculos requieren de gran precisión y exactitud, por lo que su adaptación al software de Octave resulta crucial para el uso de posteriores herramientas.

6.7.2. Explicación de la herramienta

En esta herramienta se adaptará el núcleo del simulador creado en Matla. Para comprobar su exactitud se utilizará un pequeño test con valores ya obtenidos y que deberán ser los que obtenga el software de Octave.

Por lo que para su desarrollo se adaptarán las funciones en caso de que así se requiera y se procederá a realizar el test para comprobar que sus valores son los correctos.

6.7.3. Implementación de la herramienta

La implementación de esta herramienta resulta trivial tras el desarrollo en espiral seguido a lo largo de la creación del resto de procedimientos y funciones, pues usa salvo la función principal y un método auxiliar, todas las ya implementadas anteriormente tanto de la primera sección, como de la anterior, referente al visualizado de modelos:

- ComputeLonLatIndex.m
- ComputeUVZH_IBI.m
- IBIZCurrH4.m
- LoadUVZH.m
- LoadUV_2D_IBI.m
- LoadUV_3D_IBI.m
- SimulateGlider_4DMemVTD.m
- UVZH_IBI_2_EquispacedDepth.m

- cesga_ibi_2d_netCDF_filename.m
- cesga_ibi_3d_load_depth.m
- cesga_ibi_3d_netCDF_filename.m
- cesga_ibi_load_current.m
- cesga_ibi_load_current_component.m
- cesga_ibi_load_location.m
- cesga_ibi_netCDF_close.m
- cesga_ibi_netCDF_open.m

Gracias a esto, no solo resulta trivial, sino que también es crucial a la hora de realizar las comprobaciones de todas las herramientas adaptadas y desarrolladas con anterioridad, pues se puede probar su exactitud gracias al test realizado en Matlab.

En la anterior lista de elementos se puede comprobar que la función principal del simulador es SimulateGlider_4DMemVTD , la cual no ha requerido de cambios, al igual que el nuevo método auxiliar UVZH_IBI_2_EquispacedDepth . Todo ello se debe a que todas y cada una de las funciones de la lista ya han sido adaptadas en las anteriores herramientas, por lo que no se requieren cambios en lo referente a una nueva adaptación.

Lo que sí resulta interesante comprobar es la validez del test de Octave comparado con Matlab, pues es un claro indicador de la veracidad de los resultados que se obtiene con el primer software.

El test consta de dos apartados importantes, el primero de ellos hace referencia a la función deg2km, la cual no está presente en Octave de manera nativa pero gracias a la librería externa mapping se puede tomar. Este método funciona correctamente para los mismos valores dados tanto en Matlab como en Octave.

Por otro lado, se desea realizar una prueba con el método rand. Aquí sí que se observar una diferencia clara entre Matlab y Octave, pues para la semilla 52 Matlab devuelve un valor de 0.6713 y para Octave 0.10287. Esto implicará que en determinadas herramientas que usen semillas específicas del método rand necesitarán comprobaciones específicas para Octave.

Finalmente, se tiene la comprobación del núcleo del simulador, para lo cual se ha utilizado un modelo del 27 de julio de 2014, tanto para el test realizado en Matlab como para el que he realizado en Octave, este test se desarrollará bajo la región IBI sobre la que se ha trabajado durante todo el transcurso del presente documento.

Listing 6.20: Test Octave simulador

```
cmd.interval = 20400;
cmd.bearing = 125.4061;
cmd.dLim = [10 1000];
cmd.llLim = [-18.8468 -16.6532 28.3363 30];
ll0.lon = -18.3468;
ll0.lat = 29.5848;

 %TEST
pkg load mapping;
pkg load netcdf;

printf('Primer test\n');
deg2km(1.7)

printf('Segundo test\n');
rand('seed',52)
rand

[timeS, lonS, latS, depthS, uS, vS, code] =
SimulateGlider_4DMemVTD('IBI', ll0, [2014 7 27], 0,
 0, cmd, 600, 0.2630, 0.2000, 0.1500, 1);

printf('Tercer test\n');

[timeS, lonS, latS, depthS, uS, vS, code] =
SimulateGlider_4DMemVTD('IBI', ll0, [2014 7 27], 0,
0, cmd, 600, 0.2630, 0.2000, 0.1500, 0)
```

El código de arriba se ha implementado basándome en el propio test, asignando los mismo valores con los que se realizó en el software de Matlab. Tras su ejecución se puede comprobar que ambos software, tanto Matlab como Octave devuelven los mismos valores en la ejecución del test del simulador del Glider, por lo que la prueba ha resultado exitosa, aquí se mostrará un pequeño extracto de valores obtenidos por Matlab y por Octave sobre la variable depthS:

Matlab (En base a 1.0e+03) ver figura 6.11:

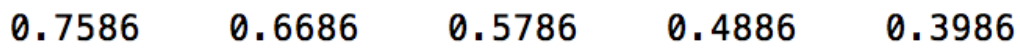

Figura 6.11: Test simulador depthS Matlab

Octave, ver figura 6.12:

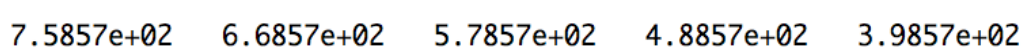

Figura 6.12: Test simulador depthS Octave

Capítulo 7

Resultados

7.1. Adaptación de netCDF en Octave

Con respecto a la adaptación de netCDF en Octave, tenemos que ésta se ha realizado con éxito, pues gracias a la librería externa de netCDF se pueden realizar las operaciones esenciales sobre archivos de esta clase, unas operaciones que no se realizan de manera nativa en Octave y que a su vez han requerido de una pequeña redefinición en las llamadas de las funciones respecto a MATLAB. Sin embargo las finalidades de las funciones son semejantes, permitiendo su uso efectivo en el resto de herramientas desarrolladas.

7.2. Indexado de modelos

En relación al indexado de modelos, se ha conseguido mejorar los dos modelos de indexado existentes, pues existían tanto campos con repetición como un cierto desorden en la forma de disponerlos. Ésto se logró solventar situando las fechas reales y de petición al principio de los ficheros, facilitando de esta manera la búsqueda por parte del investigador y elaborando una mejor disposición final de los modelos netCDF para futuros motores de búsqueda.

7.3. Gestión de proyectos

La gestión de proyectos posee dos variantes, una basada en un archivo de log que puede ser modificado directamente sin necesidad de tener conocimientos previos sobre ningún lenguaje y una segunda versión en la que prima la estandarización, usando para ello el formato XML. Se desarrollaron dos librerías destinadas a cada variante, manteniendo además la transparencia

ante el usuario, pues las funciones se llaman de la misma forma, evitando así un mayor esfuerzo para el investigador.

7.4. Adaptación del visualizado de modelos

En siguiente lugar tenemos la adaptación del visualizado de los modelos netCDF. Gracias a las funciones implementadas y adaptadas siguiendo el modelo de desarrollo en espiral, se ha logrado adaptar de Matlab a Octave la opción de poder visualizar corrientes en determinados intervalos de tiempo y de profundidad.

Como mejora adicional a la misma, se ha implementado un sistema polling para mostrar cada gráfica, permitiendo al usuario situar el puntero donde realmente desee y no tener que desplazarse desde el principio para ello, por lo que se optimiza el funcionamiento de la herramienta, siendo ésta efectiva y exitosa.

7.5. Núcleo del simulador

Con respecto al núcleo del simulador, gracias a las implementaciones anteriores ha resultado trivial la adaptación desde el software de Matlab a Octave, por lo que con el test realizado sobre el modelo del año 2014 se podía verificar que todas las herramientas desarrolladas en este trabajo funcionan correctamente, pues este núcleo hace uso de la mayoría de estas funciones.

Se tiene como resultado que se verifican los valores obtenidos por Octave, por lo que se puede concluir que el desarrollo del trabajo se ha completado con éxito.

7.6. Resultados académicos

Los resultados académicos que se han obtenido radican principalmente en la adquisición de mayores conocimientos sobre Octave, como también del manejo y tratamiento de archivos netCDF. Finalmente, he aprendido a colaborar y entender el funcionamiento de herramientas de alta relevancia para el equipo de investigación del IUSIANI.

Capítulo 8

Conclusiones

En este capítulo se tratarán las conclusiones sobre el trabajo, tanto las aportaciones como el trabajo futuro.

El conjunto de herramientas implementado y adaptado desde Matlab, permite la adaptación a un software con licencia libre como es Octave desarrollar cada una de las funcionalidades en las que el equipo de investigación se encuentra trabajando, brindando así la oportunidad de que puedan distribuir su código y modificar el software existente sin tener que estar ligado a las exclusividades del proveedor.

Por otro lado, se han desarrollado herramientas que permiten un mejor uso del sistema de directorios actual, siguiendo así un modelo centralizado que evita el mayor número de problemas que suelen poseer los sistemas distribuidos.

8.1. Aportaciones

Este trabajo aporta al Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería, un conjunto de herramientas basado en un software libre, por lo que se fomenta así el uso de dicha clase de software para el desarrollo de herramientas que apoyen la investigación.

En conclusión, el trabajo aporta herramientas no solo a la división ROCIUSIANI, sino también al sector de Investigación y Desarrollo en general, pues lograr implementar procedimientos y herramientas en un software diferente fomenta la diversidad de aplicaciones y se aumenta asimismo la capacidad de desarrollo al tener nuevas comunidades de soporte.

8.2. Trabajo futuro

Como trabajo futuro, al haber realizado un trabajo escalar y culminar con el núcleo del simulador, será posible adaptar cada uno de los simuladores en los que el equipo de investigación en Robótica y Oceanografía Computacional (ROC) del IUSIANI se encuentra trabajando, así como desarrollar nuevas herramientas basadas en archivos netCDF y Octave, pues tras la realización de este trabajo se ha descubierto que este software tiene la misma potencialidad final que Matlab para realizar este fin.

Apéndice A

Manual de Usuario

A continuación, se describirá como usar las librerías desarrolladas para la gestión de proyectos en formato texto y xml.

A.1. Creación

CREARLOG Crea un archivo LOG para listar archivos/misiones/proyectos.
Ejemplo de uso:
crearlog('ruta/prueba','archivo')

CREARMISION Crea una misión con su ruta y nombre en el LOG especificado.
Ejemplo de uso:
crearMision('rutalog','/Users/prueba/Documentos/Misiones','Mision1')

CREARPROYECTO Crea un proyecto con su ruta y nombre en el LOG especificado.
Ejemplo de uso:
crearProyecto('rutalog','/Users/prueba/Documentos/Proyectos','proyecto')

A.2. Archivos

EDITARRUTAARCHIVO Permite editar la ruta de un archivo especificado en el archivo LOG.
Ejemplo de uso:
editarRutaArchivo('rutalog','archivo','/nueva/ruta/Desktop')

GETARCHIVO Recupera la información de un archivo del LOG.
Ejemplo de uso:
archivo1 = getArchivo('rutalog', 'archivo')

A.3. Proyectos

GETPROYECTO Obtiene una estructura de datos del proyecto dada en el archivo LOG.
Ejemplo de uso:
proyecto = getProyecto('/rutalog','proyecto')

DESCRIPCIONPROYECTO Establece una descripción al proyecto del archivo LOG especificado.
Ejemplo de uso:
descripcionProyecto('rutalog','proyecto','Esto es una nueva descripcion.')

DIRECTORESPROYECTO Establecer directores en el proyecto del archivo LOG.
Ejemplos de uso:
directoresProyecto('rutalog','proyecto','Nombre1',true) SIRVE PARA ESTABLECER REEMPLAZANDO ACTUALES DIRECTORES
directoresProyecto('rutalog','proyecto','Nombre2',false) SIRVE PARA AÑADIR NUEVO DIRECTOR
directoresProyecto('rutalog','proyecto',,true) SIRVE PARA ELIMINAR TODOS LOS DIRECTORES

ELIMINARARCHIVOPROYECTO Elimina el archivo y su información de un proyecto dado en el archivo LOG.
Ejemplo de uso:
eliminarArchivoProyecto('/rutalog','proyecto','archivo')

ELIMINARMISIONPROYECTO Elimina la misión de un proyecto dado en el archivo LOG.
Ejemplo de uso:
eliminarMisionProyecto('/rutalog','Proyecto','mision')

ELIMINARPROYECTO Elimina por completo el proyecto dado del archivo LOG.
Ejemplo de uso:
eliminarProyecto('rutalog','proyecto')

FECHAFINALPROYECTO Establece la fecha final en formato (AAAA/MM/DD) del proyecto del archivo LOG.
Ejemplo de uso:
fechaFinalProyecto('/rutalog','proyecto','1996/06/04')

FECHAINICIOPROYECTO Establece la fecha de inicio del proyecto con formato AAAA/MM/DD del archivo LOG.
Ejemplo de uso:
fechaInicioProyecto('/rutalog','proyecto','04/06/1996')

INCLUIRARCHIVOPROYECTO Añade un archivo en un proyecto del archivo LOG.
Ejemplo de uso:
incluirArchivoProyecto('/rutalog','proyecto','archivo','IBISR','HYCOM') Se debe colocar la ruta completa SIN su extensión.

INCLUIRMISIONPROYECTO Incluye una misión existente en el archivo LOG en el proyecto dado.
Ejemplo de uso:
incluirMisionProyecto('/rutalog','proyecto','mision')

PARTICIPANTESPROYECTO Establecer participantes en el proyecto del archivo LOG.
Ejemplos de uso:
participantesProyecto('rutalog','proyecto','Nombre1',true) SIRVE PARA ESTABLECER REEMPLAZANDO ACTUALES PARTICIPANTES
participantesProyecto('rutalog','proyecto','Nombre2',false) SIRVE PARA AÑADIR NUEVO DIRECTOR
participantesProyecto('rutalog','proyecto',,true) SIRVE PARA ELIMINAR TODOS LOS PARTICIPANTES

PUBLICACIONESPROYECTO Establecer publicaciones en el proyecto del archivo LOG.
Ejemplos de uso:
publicacionesProyecto('rutalog','proyecto','Nombre1',true) SIRVE PARA ESTABLECER REEMPLAZANDO ACTUALES PUBLICACIONES
publicacionesProyecto('rutalog','proyecto','Nombre2',false) SIRVE PARA AÑADIR NUEVA PUBLICACION
publicacionesProyecto('rutalog','proyecto',,true) SIRVE PARA ELIMINAR TODAS LAS PUBLICACIONES

RENOMBRARPROYECTO Renombra un proyecto dado en el archivo LOG
Ejemplo de uso:
renombrarProyecto('/rutalog','proyecto','proyecto')

RUTAPROYECTO Establece la ruta del proyecto dado en el archivo LOG.
Ejemplo de uso:
rutaProyecto('/rutalog','proyecto','/Ruta/New')

A.4. Misiones

GETMISION Obtiene una estructura de datos de la misión dada en el archivo LOG.
Ejemplo de uso:
mision = getMision('rutalog','mision')

DESCRIPCIONMISION Establece la descripción de una misión en el archivo LOG.
Ejemplo de uso:
descripcionMision('rutalog','mision1','Nueva descripcion.')

DIRECTORESMISION Establecer directores en la misión del archivo LOG.
Ejemplos de uso:
directoresMision('rutalog','mision1','Nombre1',true) SIRVE PARA ESTABLECER REEMPLAZANDO ACTUALES DIRECTORES
directoresMision('rutalog','mision1','Nombre2',false) SIRVE PARA AÑADIR NUEVO DIRECTOR
directoresMision('rutalog','mision1',,true) SIRVE PARA ELIMINAR TODOS LOS DIRECTORES

ELIMINARARCHIVOMISION Elimina el archivo y su información de una misión dada en el archivo LOG.
Ejemplo de uso:
eliminarArchivoMision('rutalog','mision','archivo')

ELIMINARMISION Elimina una misión dada del archivo LOG.
Ejemplo de uso:
eliminarMision('rutalog','mision1')

FECHAFINALMISION Establece la fecha final de la misión en formato AAAA/MM/DD en el archivo LOG.
Ejemplo de uso:
fechaFinalMision('rutalog','mision','1996/04/06')

FECHAINICIOMISION Establece la fecha inicio de la misión en formato AAAA/MM/DD en el archivo LOG.
Ejemplo de uso:
fechaInicioMision('rutalog','mision','1996/04/06')

INCLUIRARCHIVOMISION Añade un archivo en una misión del archivo LOG.
Ejemplo de uso:
incluirArchivoMision('rutalog','mision1','archivo', 'IBISR','HYCOM') Se debe colocar la ruta completa SIN su extensión.

PARTICIPANTESMISION Establecer participantes en la misión del archivo LOG.
Ejemplos de uso:
participantesMision('rutalog','mision1','Nombre1',true) SIRVE PARA ESTABLECER REEMPLAZANDO ACTUALES PARTICIPANTES
participantesMision('rutalog','mision1','Nombre2',false) SIRVE PARA AÑADIR NUEVO PARTICIPANTE
participantesMision('rutalog','mision1',,true) SIRVE PARA ELIMINAR TODOS LOS PARTICIPANTES

PUBLICACIONESMISION Establecer publicaciones en la misión del archivo LOG.
Ejemplos de uso:
publicacionesMision('rutalog','mision1','Nombre1',true) SIRVE PARA ESTABLECER REEMPLAZANDO ACTUALES PUBLICACIONES
publicacionesMision('rutalog','mision1','Nombre2',false) SIRVE PARA AÑADIR NUEVA PUBLICACION
publicacionesMision('rutalog','mision1',,true) SIRVE PARA ELIMINAR TODOS LAS PUBLICACIONES

RENOMBRARMISION Establece un nuevo nombre de misión en el archivo LOG.
Ejemplo de uso:
renombrarMision('rutalog','mision','nuevaMision')

RUTAMISION Establece la ruta de la misión dada en el archivo LOG.
Ejemplo de uso:
rutaMision('rutalog','mision','/ruta2')

Bibliografía

[1] Marine Copernicus CMEMS. Copernicus Marine Environment Monitoring Service.

[2] Marine Copernicus CMEMS. Copernicus Marine Environment Monitoring Service, Glossary Terms.

[3] Copernicus. *Copernicus IBI Product User Manual.* Marcos G. Sotillo, Guillaume Reffray, Arancha Amo, Bruno Levier, 2017.

[4] COMISION EUROPEA. *Data Warehouse Requirements 2.0 - Copernicus Data Access - Specifications of the space-based Earth Observation needs for the period 2014-2020.* 2014.

[5] COMISION EUROPEA. *Technical Annex to the Delegation Agreement with Mercator Océan for the implementation of the Copernicus Maritime Service.* 2014.

[6] COMISION EUROPEA. *Technical Annex for the budget implementation tasks linked to the provision of the Copernicus Atmosphere Monitoring Service (CAMS).* 2016.

[7] COMISION EUROPEA. *Technical Annex for the budget implementation tasks linked to the provision of the Copernicus Climate Change (C3) service.* 2016.

[8] Octave. About octave.

[9] W3 Schools. XML Validator.

[10] Tuxtrans. Why you should use free software.

[11] Unidata UCAR. Netcdf Data Types, The netCDF Interface.

[12] Unidata UCAR. Netcdf DataSet Components, The netCDF Interface.

[13] Unidata UCAR. An Introduction to NetCDF, The netCDF Interface.