Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE FI DE CARRERA

TÍTOL: Aplicación informática para la integración de los datos generados por el observatorio OBSEA en las redes de sistemes de observación.

AUTOR: Alejandro Borrell González

TITULACIÓ: Ingeniería Técnica en Informática de Gestión

DIRECTOR: Antoni Mànuel Làzaro

DEPARTAMENT: Ingeniería Electrónica

PONENT: Rafael Morillas Varón

DEPARTAMENT: Ingeniería Telemática

DATA: Junio de 2012

TÍTOL: Aplicación informática para la integración de los datos generados por el observatorio OBSEA en las redes de sistemes de observación.

COGNOMS: Borrell González **NOM: Alejandro**

TITULACIÓ: Ingeniería Técnica en Informática

ESPECIALITAT: Gestión **PLA: 92**

DIRECTOR: Antoni Mànuel Làzaro

DEPARTAMENT: Ingeniería Electrónica

PONENT: Rafael Morillas Varón

DEPARTAMENT: Ingeniería Telemática

QUALIFICACIÓ DEL PFC

TRIBUNAL

PRESIDENT **SECRETARI** **VOCAL**

DATA DE LECTURA: 2 de Julio de 2012

Aquest Projecte té en compte aspectes mediambientals: Sí X No

PROJECTE FI DE CARRERA

RESUM (màxim 50 línies)

En mayo de 2009 se puso en funcionamiento el primer observatorio submarino cableado en España llamado OBSEA, que transmite datos recolectados del suelo marino en tiempo real a los laboratorios de sus dos creadoras (CSIC y UPC).

El principal objetivo de este proyecto es el desarrollo de una aplicación informática capaz de manipular y transformar los datos recibidos por el observatorio para que estos sean entendibles e interpretables por parte de toda la comunidad científica, con el fin de conseguir una mayor difusión de los mismos.

Para ello se necesita de una aplicación que cumpla con los requisitos necesarios para que se adapte a los diferentes tipos de datos, procedentes de los diferentes instrumentos que componen el observatorio, que se reciben y sea capaz de detectar, de forma automática, el tipo de dato que obtenemos en todo momento. Además, deberá tener una interfaz simple para facilitar la utilización de la aplicación al usuario.

Con el fin de cumplir el objetivo principal de este proyecto, una vez se haya conseguido la creación de archivos contenedores de la información procedente del observatorio en formato estándar, se realizará la difusión de dicha información. Para ello, será necesario registrar el observatorio OBSEA en la red de sistemas de observación SeaDataNet, donde se irán añadiendo los archivos creados con la información que deseemos con el fin de que nuestros datos puedan ser consultados por el resto de usuarios, centros o laboratorios.

Paraules clau (màxim 10):

NetCDF	OBSEA	Aplicación	MySQL
SeaDataNet	Datos Oceanográficos	Java	Vocabularios Estandarizados
CF-Conventions			

1. Introducción

En 2009 se puso en funcionamiento el primer observatorio submarino cableado del estado español OBSEA (Observatorio Submarino Expandible), un proyecto fruto de la iniciativa conjunta del CSIC[1]/UPC.

OBSEA transmite en tiempo real a los centros de investigación series de datos recogidos del fondo marino del mar Mediterráneo, lo que permite estudiarlo a través de Internet.

Actualmente estos datos recibidos por el observatorio se almacenan en los servidores del centro de desarrollo SARTI (Sistemas de Adquisición Remota y Tratamiento de la Información), de la Universidad Politécnica de Cataluña, y no tienen toda la difusión que sería de esperar. Por eso, el centro, como unidad colaboradora en el diseño y la construcción de la plataforma submarina, ha tomado la decisión de integrar dichos datos en las redes de sistemas de observación existentes.

Para ello, en primer lugar se modelarán y transformarán los datos, generados por OBSEA, obteniendo archivos que cumplirán una serie de estándares y podrán ser visualizados e interpretados por diferentes herramientas software.

Una vez obtenidos este tipo de archivos, como objetivo principal de este proyecto, se desea compartirlos con el resto de la comunidad. Concretamente en **SeaDataNet**, una red de sistemas de observación.

Figura 1.1: Observatorio OBSEA

[1] CSIC: Consejo Superior de Investigación Científica

1.1 OBSEA: Observatorio Submarino Expandible

OBSEA es un observatorio submarino situado a unos 4 km de la costa de Vilanova i la Geltrú en una zona protegida de pesca. Está interconectado a la costa por un cable mixto de energía y comunicaciones. El objetivo del proyecto OBSEA es la creación de una red de nodos submarinos en los que poder conectar diferentes sensores de acuerdo con las necesidades de los científicos en cada momento. En cada uno de estos nodos se pueden conectar hasta 8 instrumentos diferentes. Inicialmente, en el primero de los nodos, se dispone de una cámara IP con movimiento, un CTD[2] (Conductivity, Temperature, Depth) para medir la conductividad, la temperatura y profundidad, un hidrófono para medir las diferentes emisiones acústicas que podemos encontrar en el mar Mediterráneo, ya bien sean de origen natural o provocadas por el ser humano, y de una boya en la superficie ubicada a 40 metros del OBSEA. Esta, además de indicar la posición en la que se encuentra el observatorio, es una extensión del mismo que funciona como una plataforma útil para realizar medidas oceanográficas y de parámetros ambientales, siendo además un banco de pruebas para sensores marinos, equipamiento e instrumentos oceanográficos donde se aprovechan las ventajas de la infraestructura y la conectividad del OBSEA.

La principal ventaja que aporta el observatorio OBSEA es la de poder obtener datos, de los instrumentos instalados en cada momento, a tiempo real, las 24 horas del día durante todos los días del año. De esta forma, se pueden almacenar los datos recibidos para, posteriormente, analizar eventos singulares que se puedan haber producido.

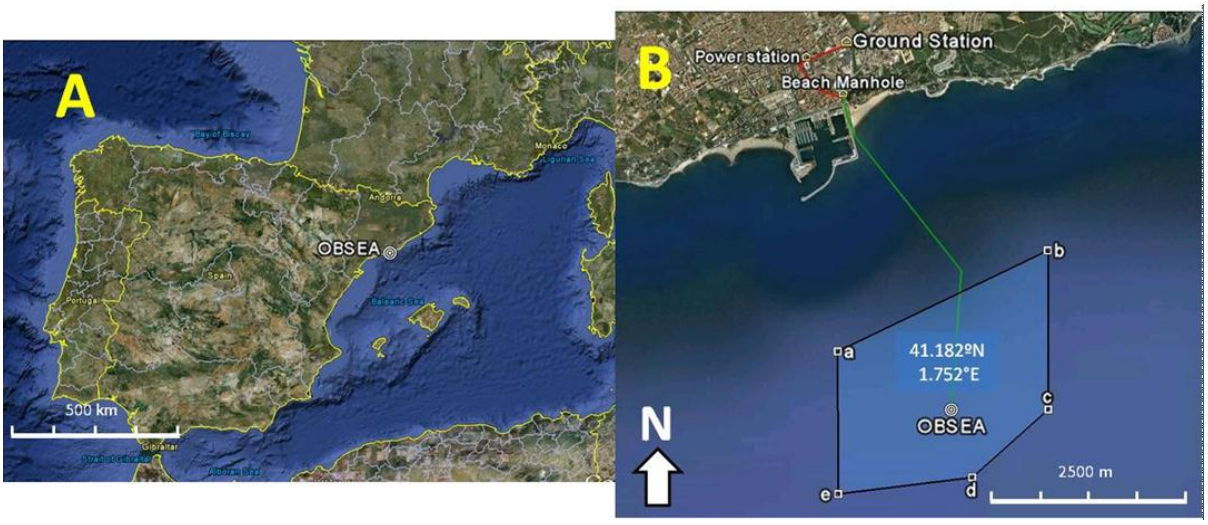


Figura 1.2: Localización del observatorio OBSEA

[2] CTD: Instrumento para la medición de Conductividad, Temperatura y Profundidad (Depth)

1.2 Objetivo del proyecto

Los datos generados por el observatorio OBSEA se almacenan en la Base de Datos del servidor del Centro de Desarrollo SARTI y no tienen toda la difusión que sería de esperar. Por esta misma razón, es necesario modelar y transformar los datos, actualmente en formato de texto (ASCII), en una Base de Datos MySQL, a formato NetCDF (**Net**work **C**ommon **D**ata **F**orm), con la información que tenemos almacenada procedente de los CTD's del observatorio, los cuales nos proporcionan datos de temperatura, conductividad, salinidad, presión y velocidad del sonido debajo del agua.

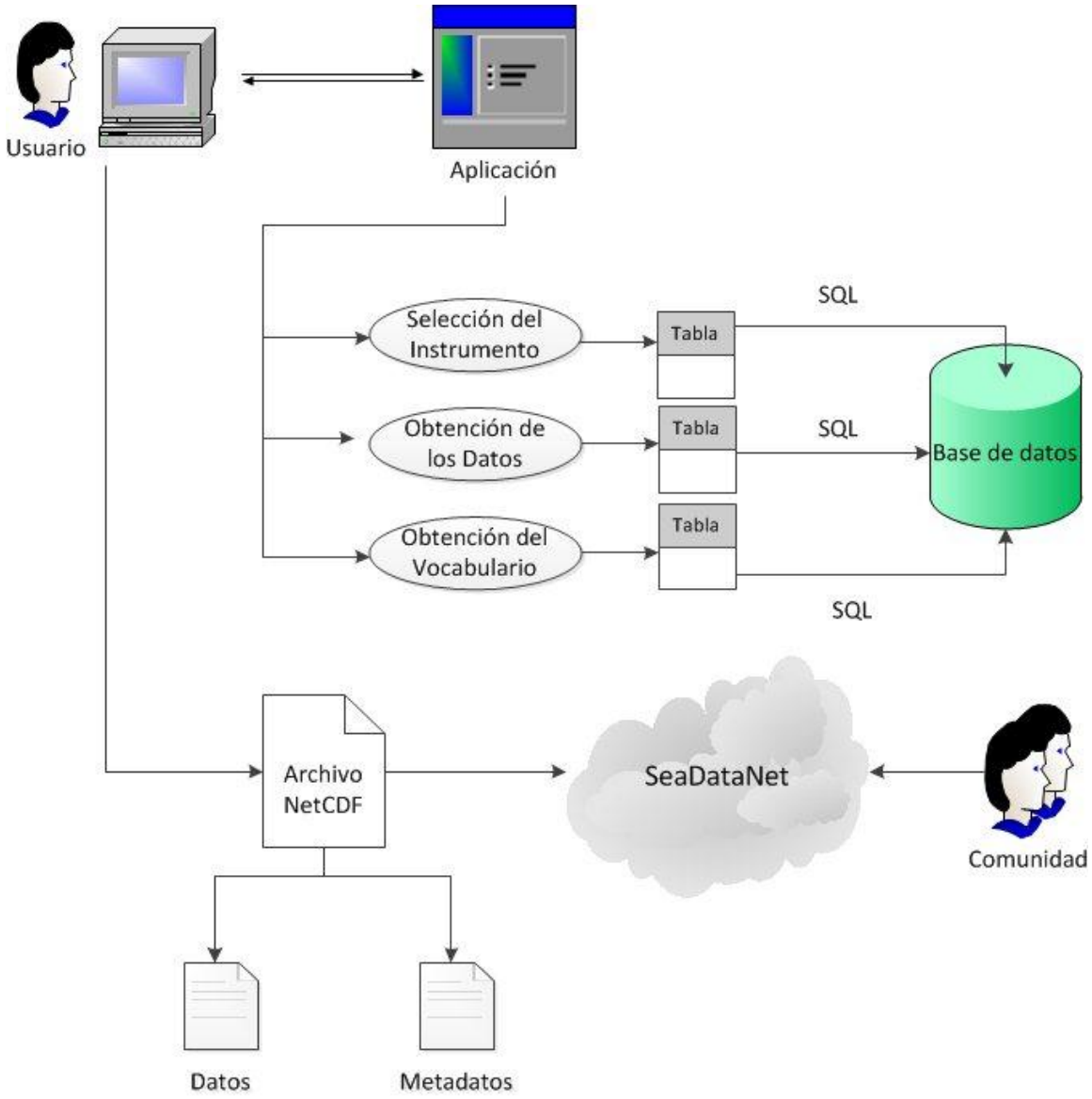


Figura 1.3: Diagrama general del proyecto

Para la transformación de los datos almacenados en la Base de Datos se desarrollará una aplicación en lenguaje JAVA, utilizando la API de Java para NetCDF (*NetCDF-Java Library version 4.2*), para la fácil creación de este tipo de archivos mediante una interfaz simple e intuitiva para facilitar la tarea al usuario.

Además de los datos proporcionados por los CTD's que hemos mencionado antes, los archivos contendrán información adicional, los llamados metadatos, como por ejemplo la ubicación exacta del observatorio (latitud, longitud y profundidad), el nombre del centro u organización del cual procede dicha información, etc. para una fácil y rápida interpretación de los datos.

Toda esta información contenida dentro de los NetCDF, deberá cumplir una normativa a nivel de vocabulario, normas establecidas según la "Climate and Forecast Metadata Conventions" versión 1.6 (la más actual), para que después estos archivos puedan ser interpretados por software como el ODV (Ocean Data View) donde los datos pueden ser visualizados de forma gráfica.

Con los archivos ya creados, el siguiente paso será integrarlos en la red de observación SeaDataNet, donde registrando y dando de alta en el sistema el observatorio OBSEA, podremos ir integrando los datos generados para que sean accesibles y visualizados por cualquier persona, centro o laboratorio mediante una interfaz estándar.

La creación de una red de sistemas de observación como SeaDataNet resuelve la problemática que normalmente se encuentra el usuario cuando llega el momento de visualizar o interpretar archivos con diferentes tipos de información. La finalidad es crear un modelo estándar de datos, es decir, una única vía común de almacenamiento, ya que de lo contrario se dificulta sensiblemente su difusión.

De esta situación parte la iniciativa de SeaDataNet, donde se pretender dar una solución al problema estableciendo una normativa a los archivos que cada centro desee compartir con la comunidad, cumpliendo una serie de requisitos y conteniendo información sobre ellos mismos (metadatos), para facilitar su interpretación.

2. NetCDF (Network Common Data Form)

Es un conjunto de librerías software, creadas por UNIDATA, que soporta la creación, el acceso y uso compartido de conjuntos (archivos o "*dataset*") de datos científicos. NetCDF cumple con el objetivo de CDI (Common Data Index) y está basado en el estándar de metadatos[3] ISO 19115, motivos importantes por el que se ha escogido este formato de archivo.

2.1 Unidata

El formato de archivo NetCDF (Network Common Data Form) fue implementado por el programa Unidata, uno de los ocho programas de la UCAR[4], y establecido como formato estándar para la comunidad científica para el almacenamiento de todo tipo de datos oceanográficos y atmosféricos.

Unidata es una comunidad formada por más de 250 instituciones reconocidas con una misma meta común, la del intercambio de datos y herramientas para el acceso y la visualización de datos. Por ello, esta comunidad nos proporciona una gran variedad de servicios y herramientas para poder acceder y manipular una gran parte de los tipos de datos existentes en la actualidad.

El **Unidata Program Center**, como líder de una amplia comunidad:

- Explora nuevas tecnologías
- Evalúa e implementa los estándares tecnológicos y herramientas
- Desarrolla soluciones innovadoras y nuevas capacidades para resolver las necesidades de la comunidad.
- Facilita la búsqueda de datos y uso de las bibliotecas digitales
- Los valores con los estándares abiertos, interoperabilidad, y los enfoques en código "*open-source"*.
- Permite a los estudiantes el aprendizaje centrado en las ciencias mediante el fomento de la utilización de datos y herramientas en la educación.

[3] Metadatos: Datos acerca de datos.

[4] UCAR: University Corporation for Atmospheric Research

- Mantenerse informado de las tendencias de computación, ya que pertenecen a la investigación avanzada y la educación.

2.2 Estándares que cumple NetCDF

2.2.1 OGC (Open Geospatial Consortium)

UCAR presentó NetCDF en el OGC (Open Geospatial Consortium) como candidato a estándar OGC para fomentar una mayor utilización internacional y una mayor interoperabilidad entre clientes y servidores que intercambian datos.

El **O**pen **G**eospatial **C**onsortium (OGC) es una organización internacional sin ánimo de lucro. Su fin es la definición de estándares abiertos e interoperables dentro de los Sistemas de Información Geográfica.

Persigue acuerdos entre las diferentes empresas del sector que posibiliten la interoperación de sus sistemas de geo procesamiento y facilitar el intercambio de la información geográfica en beneficio de los usuarios. Anteriormente fue conocido como "Open GIS Consortium" debido a que antes de que firmara como *Consortium* estuvo registrada como una fundación.

2.2.2 CDI (Common Data Index)

El objetivo principal del "Common Data Index" es dar a los usuarios una visualización muy detallada, como a su vez de una disponibilidad y gran difusión geográfica, de información oceanográfica a través de los datos obtenidos de diferentes instituciones de toda Europa. El CDI ofrece un índice (los metadatos) a los conjuntos de datos individuales, basados en el estándar ISO 19115 mencionado anteriormente, que facilita el camino al usuario cuando quiere acceder a dichos datos.

2.2.3 ISO 19115

La ISO (*International Organization for Standardization*), es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación. Su función principal es buscar la estandarización de normas de productos para organizaciones a nivel internacional.

En concreto, la ISO 19115 define el esquema requerido para describir información geográfica. Proporciona información sobre la identificación, amplitud, calidad, esquema espacial y temporal y distribución de datos geográficos.

ISO 19115 es aplicable a:

- La catalogación de conjuntos de datos.
- Actividades de intercambio de información.
- Descripción completa de conjuntos de datos geográficos.
- Propiedades y características de los datos.

ISO 19115 define:

- Como obligatorios y condicionales secciones de metadatos.
- El conjunto mínimo de metadatos requeridos.
- Los elementos de metadatos opcionales para permitir una descripción normalizada más amplia de los datos.
- El método para la extensión de los metadatos en caso de la necesidad de adaptaciones específicas.

Aunque la norma ISO 19115 es aplicable a los datos digitales, sus principios pueden extenderse a muchas otras formas de datos geográficos, tales como mapas, cartas y documentos textuales, así como datos no geográficos.

2.3 Características de los Archivos NetCDF

La principal característica de este tipo de archivos es que no necesitan de un archivo adicional (como por ejemplo un archivo adjunto XML) para su interpretación. Son archivos que se describen y se explican a sí mismos (self-describing), lo que facilita su visualización por parte del usuario.

En él, está contenida suficiente información como para saber qué clase de datos se encuentran dentro de el mismo: tipo de variables, tipos de datos, unidades, institución creadora, etc…

Para que toda esta información sea interpretada utilizaremos el vocabulario estándar especificado en la "*NetCDF Climate and Forecast Metadata Conventions*".

2.3.1 Climate and Forecast (CF) Metadata Conventions

Este documento describe el vocabulario específico que deberemos usar para cada uno de nuestros archivos NetCDF, dependiendo de qué tipo de variables y datos haya contenidos en él, de metadatos diseñados para la promoción, transformación e intercambio de archivos creados con interfaz de programación de aplicaciones NetCDF.

Este convenio (en nuestro caso el CF-1.6) define los metadatos para que nos proporcionen una descripción definitiva de los datos que representan cada variable de nuestro archivo, de la distribución espacial y de las propiedades temporales de los datos. Nos da una definición precisa de cada variable a través de la especificación de un nombre estándar para ella, nos indica las coordenadas y la dimensión (espacio/tiempo) a las que corresponde dicha variable.

2.3.2 Estructura de los Archivos NetCDF

Los archivos NetCDF tienen una estructura interna definida por el CDM (***C****ommon* ***D****ata* ***M****odel*).

El "*Common Data Model"* de Unidata es un modelo estandarizado de datos abstractos para conjuntos de datos científicos. Este modelo fusiona los modelos NetCDF, OPeNDAP y los modelos HDF5 para crear una API común para muchos tipos de datos científicos. La librería NetCDF de Java es una implementación del CDM, la cual puede leer varios formatos de archivos además del propio NetCDF. Les llamamos archivos

CDM, una abreviatura para los archivos que se pueden leer con las librerías NetCDF de Java y acceder a ellos a través del modelo de datos CDM.

El CDM consta de varias capas, donde cada una se va superponiendo en la parte superior de la anterior, para ir añadiendo cada vez una semántica más compleja:

1. La **capa de acceso de datos**, se encarga de los datos de lectura y escritura.
2. La **capa de sistemas de coordenadas**, identifica las coordenadas de las matrices de datos.
3. La **capa de atributo estándar** es la que conoce algunos de los significados que los humanos utilizamos para hacer referencia a los datos científicos tales como unidades, sistemas de coordenadas, topología de datos, etc…
4. La **capa de características de tipo de dato científico**, identifica los tipos de datos añadiendo métodos especializados para cada tipo de datos.

El siguiente diagrama muestra la estructura interna de cualquier archivo definido por el *Common Data Model*.

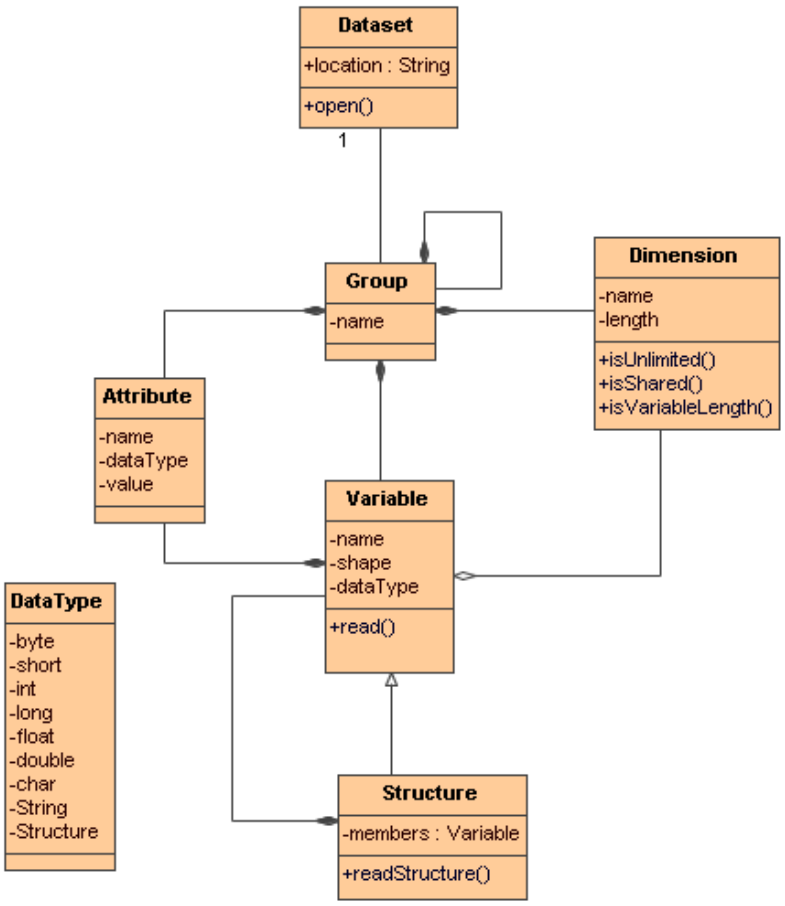


Figura 2.1: Diagrama UML de la estructura básica de CDM

Una vez vistas las capas anteriores, las cuales nos enseñan la estructura general de cualquier archivo CDM (en nuestro caso NetCDF), vamos a ver algunos de los tipos de datos abstractos más importantes que las componen:

1. **Conjunto de Datos (Dataset):** Puede ser un NetCDF, HDF5, OPeNDAP o cualquier otro tipo de documento al que se pueda acceder a través de la API de NetCDF.

2. **Grupo:** Un grupo contiene variables, dimensiones, atributos, etc… En definitiva, los grupos son el contenedor de todo aquello que contiene nuestro "*dataset"* formando un árbol jerárquico. Por eso, como mínimo siempre habrá un grupo definido en nuestro archivo.

3. **ProtoVariable:** En los archivos NetCDF una ProtoVariable es un contenedor de datos. Está compuesta por varias variables que contienen su información, como por ejemplo el tipo de datos que contiene, su dimensión o dimensiones que definen su matriz y, opcionalmente, se pueden complementar añadiéndole un conjunto de atributos que la describan o definan aun con mayor exactitud.

4. **Dimensión:** Una dimensión se utiliza para definir en función de qué van a variar los valores de nuestra variable. Puede ser compartida entre las demás variables lo que proporciona una manera simple y eficaz de asociarlas.

5. **Atributo:** Un atributo tienen un nombre y un valor que se utilizan para asociar metadatos con una variable o grupo. Por ejemplo, si se trata de un atributo de una variable estaríamos hablando de unidades o nombre estándar y solamente afectaría a esta variable. Si nos referimos a un atributo de un grupo, este estaría afectando a todo el conjunto de datos y el atributo sería, por ejemplo, para definir qué convención se ha utilizado o el nombre de la organización que ha generado estos datos.

6. **Estructura:** Es un tipo de variable que contiene otras variables. De esta manera los datos almacenados en una estructura se encuentran físicamente muy juntos en el disco, lo que hace que sea eficaz la recuperación de estos datos al mismo tiempo.

7. **Array:** Un array (o matriz) contiene los datos reales de una variable obtenidos del disco, red, base de datos, etc. Podemos decir que un array es el contenedor de las series de datos.

3. Herramientas utilizadas

3.1 NetBeans IDE 7.0.1

NetBeans es un entorno de desarrollo integrado (Integrated Development Enviroment con abreviación IDE) libre y gratuito, una herramienta para el desarrollo de aplicaciones, especialmente para Java aunque sirve perfectamente para cualquier otro tipo de lenguajes de programación como C y C++, mediante el cual se pueden crear aplicaciones gráficas.

Esta plataforma permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes software llamados módulos. En este caso, un módulo es un archivo Java (extensión .java) que contiene clases Java escritas para interactuar con la API de NetBeans. Las aplicaciones que son construidas a partir de estos módulos pueden ser extendidas fácilmente por otros desarrolladores de software agregándoles nuevos módulos.

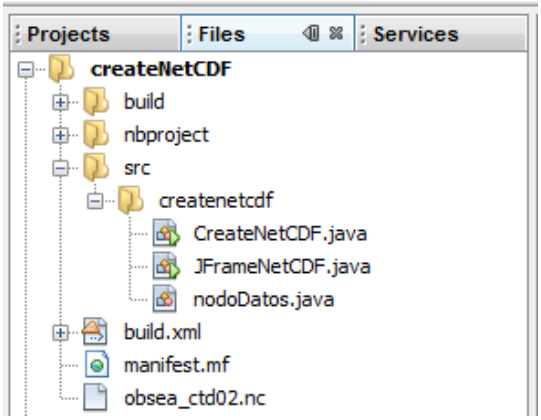


Figura 3.1: Estructura del proyecto

Esta herramienta nos permite gestionar, de una manera fácil, todos los aspectos relacionados con el desarrollo de aplicaciones, desde la creación de clases hasta el diseño de la interface de nuestra aplicación.

En la parte lateral del programa podemos ver la estructura completa de nuestra aplicación con los distintos archivos y directorios de esta. Desde este menú situado a la izquierda de la pantalla, podemos crear nuevos módulos o modificar los ya existentes.

Respecto a la programación, NetBeans nos proporciona muchas facilidades con el código. Nos facilita la inserción de éste haciendo el desarrollo de la aplicación mucho más rápida y ágil. Nos indica errores de sintaxis que haya en el código y posibles alternativas para solucionar dichos errores.

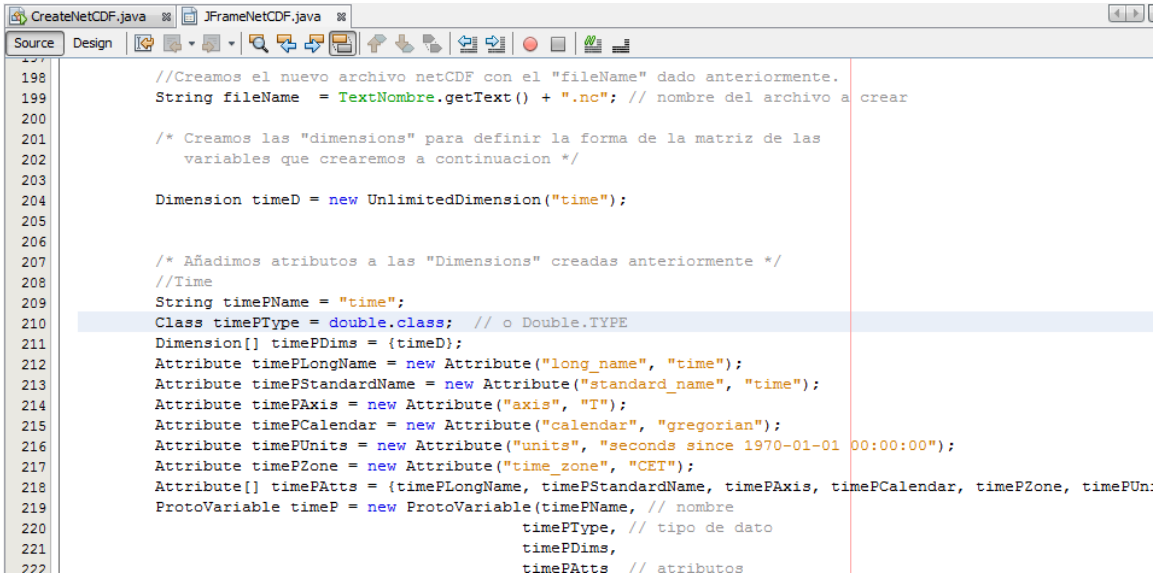


Figura 3.2: Área de trabajo de NetBeans IDE

En la parte inferior del área de trabajo de NetBeans nos encontramos con el recuadro de salida o "console" donde el IDE nos mostrará posibles errores que se hayan producido en la compilación y ejecución de nuestra aplicación o el resultado de esta. En el caso de producirse algún error, nos mostrará un mensaje, de color rojo, indicando que se ha producido algún error y en qué línea se ha producido exactamente, lo que facilita muchísimo la corrección de errores de compilación, ya que podemos estar desarrollando aplicaciones de miles de líneas de código.

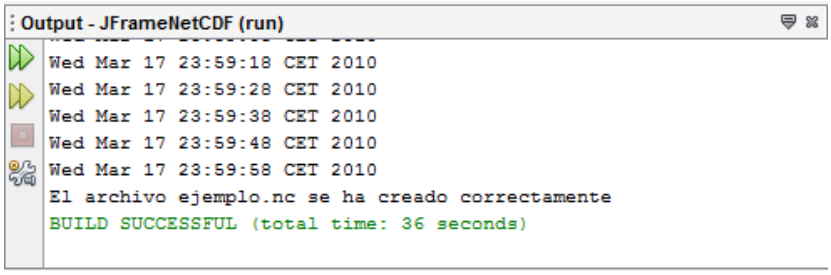


Figura 3.3: Consola

De no producirse ningún error, nos mostrará por pantalla un mensaje de texto, de color verde, conforme la compilación y ejecución del programa se ha producido de forma satisfactoria.

NetBeans también es un entorno de desarrollo potente cuando llegue el momento de hacer el diseño para nuestra aplicación. Dispone de un menú lateral con una amplia selección de todo tipo de elementos (paneles, botones, etiquetas, menús, etc…) para cubrir cualquier necesidad de nuestra aplicación. Para su utilización solamente deberemos seleccionar el elemento, arrastrarlo a nuestro marco de trabajo y colocarlo en la zona que nosotros deseemos.

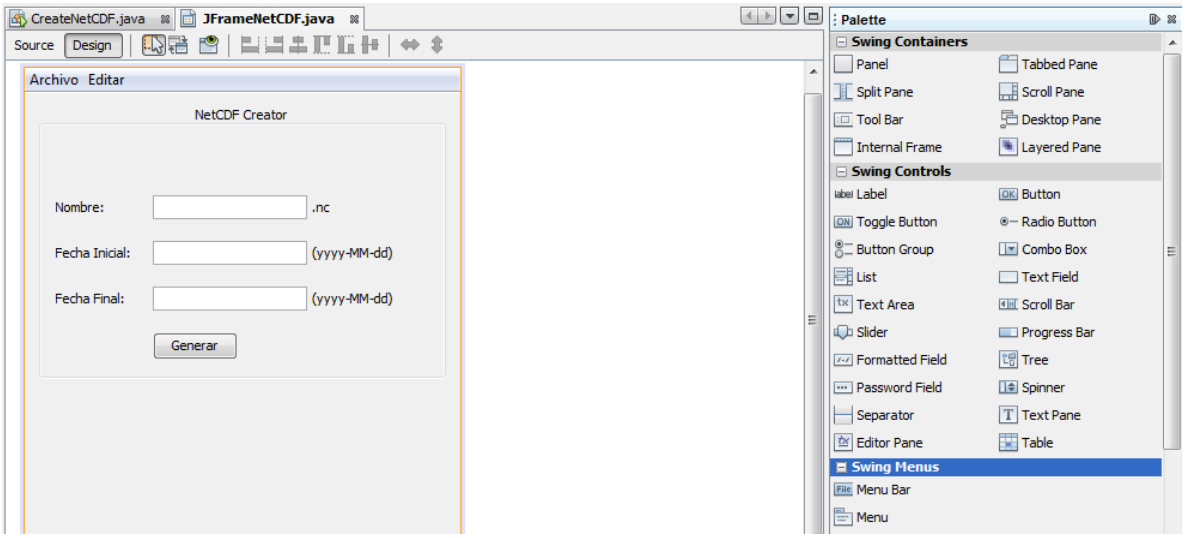


Figura 3.4: Área de diseño

3.2 Servidor XAMPP

XAMPP es un servidor independiente de plataforma que consiste en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. El nombre proviene del acrónimo de **X** (cualquier sistema operativo), **A**pache, **M**ySQL, **P**HP, **P**erl.

El paquete XAMPP (**X** refiriéndose a cualquier Sistema Operativo, **A**pache, **M**ySQL, **P**HP, **P**erl) es un conjunto de aplicaciones que tienen el objetivo de hacer la instalación y configuración de un servidor web, un proceso rápido y sencillo. El paquete está compuesto de las siguientes aplicaciones:

- **Servidor Web Apache 2.2.21**
- **Servidor MySQL 5.5.16**
- **Servidor PHP 5.3.8**
- **FileZilla FTP Server 0.9.41**
- **Mercury Mail**

XAMPP dispone de un panel de control donde, en cada momento y según nuestras necesidades, podemos activar y desactivar las aplicaciones y servicios que en cada momento nos hará falta utilizar.

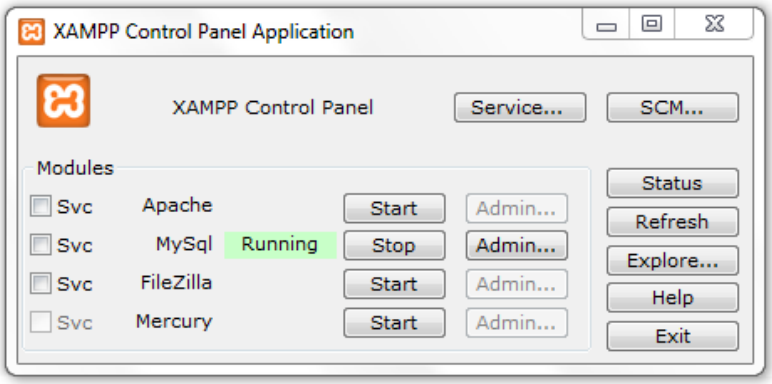


Figura 3.5: Panel de control de XAMPP

En este caso solamente utilizaremos el servidor MySQL, un sistema de gestión de bases de datos donde se almacenarán todos los datos que utilizaremos a lo largo del proyecto.

Esta aplicación se utilizará para crear o modificar las tablas que requiera el proyecto, todo esto en una base de datos local, en mi propio ordenador, mientras dure el desarrollo de la aplicación para ir comprobando su funcionamiento. Una vez finalizado y se hayan hecho las pruebas necesarias con éxito que garanticen su total funcionamiento, se añadirán o modificarán las tablas que sean necesarias en la base de datos del grupo SARTI donde se alojarán finalmente.

Además de los servicios mencionados anteriormente, también se incluye un gestor llamado phpMyAdmin, para la configuración de MySQL muy sencillo. Por eso, para realizar esta tarea se utiliza otra herramienta más potente y completa que se explicará más adelante en el siguiente apartado.

3.3 Gestor de Bases de datos: SQL Manager for MySQL

Para a gestión de las bases de datos, como ya he dicho anteriormente, la herramienta que estaba incluida en paquete XAMPP, phpMyAdmin, resultaba demasiado simple, con limitadas opciones y con una interfaz poco intuitiva y poco agradable ya que el diseño es bastante simple y eso le hace parecer una herramienta antigua. Por todos estos motivos y porque las bases de datos tendrán un papel muy importante durante el proyecto, he preferido escoger otro gestor para bases de datos MySQL llamado **SQL Manager** for MySQL.

Este programa es una herramienta dedicada exclusivamente a la gestión de bases de datos MySQL, su utilidad es mucho mayor que la de la herramienta descartada, ya que dispone de muchísimas más opciones, por lo que la convierte en una herramienta mucho más completa y robusta. Aunque disponga de una gran variedad de funciones sigue siendo una aplicación muy fácil de usar e intuitiva y su interfaz es mucho más agradable desde el punto de vista del usuario.

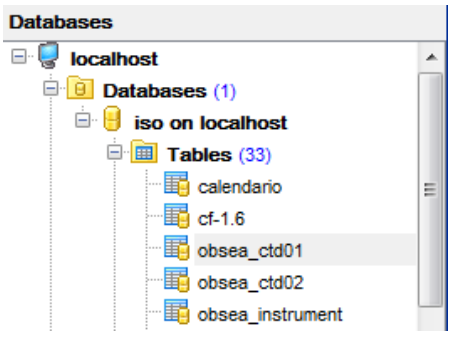


Figura 3.6: Lista de tablas

En la parte izquierda del gestor, podemos ver el listado de tablas que contiene nuestra base de datos, en este caso todas ellas alojadas en "localhost". Para crear nuevas tablas y añadirlas a nuestra base de datos el proceso será bastante sencillo, ya que el asistente nos guiará a lo largo de todo el proceso.

El primer paso que se nos pedirá es que asignemos un nombre a nuestra nueva tabla (*figura 3.7*) seguidamente, en la segunda pestaña, deberemos añadir todos los campos que necesitemos en nuestra tabla (*figura 3.8*) a los que también se les asignará un nombre, el tipo de datos que contendrán, su longitud máxima, si se tratará de una clave primaria (campo con un valor diferente para cada uno de los registros que se irán añadiendo) o si se quiere evitar que a ese campo se le asigne un valor nulo en caso de que no contenga información. Una vez hayamos insertados todos nuestros campos compilaremos la tabla para que se almacene en nuestra base de datos. Al finalizar la creación de la tabla, esta podrá ser modificada tantas veces como nosotros queramos o requiera el momento eliminando o añadiendo nuevos campos dentro de esta.

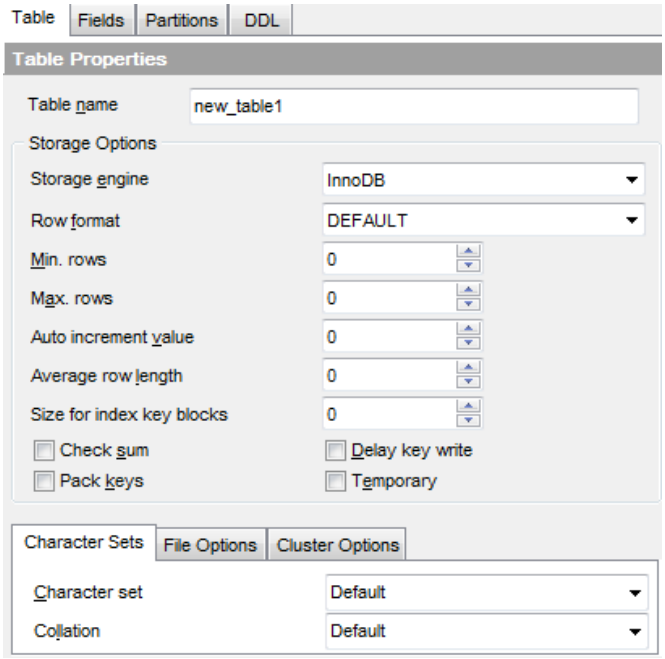


Figura 3.7: Creación de una tabla

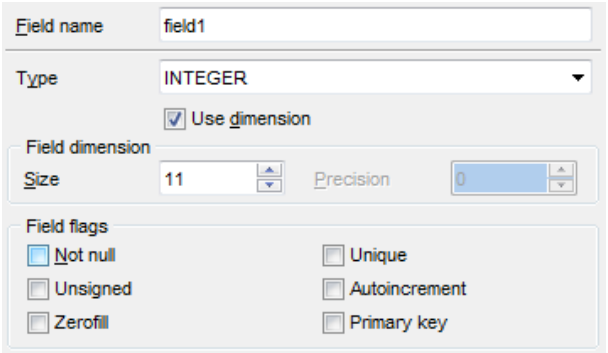


Figura 3.8: Creación de un campo

Una vez creado o creados nuestros campos, estos se añadirán a la tabla y ésta los mostrará en una lista "Fields" donde podremos ver sus principales características (*figura 3.9*). A través de esta lista, la aplicación nos ofrece la opción de crear nuevos campos o modificar o eliminar los que ya existen.

Properties | Fields | Indices | Triggers | Data | Dependencies | DDL

Field Name	Field Type	Size	Precision	Not Null	Unsigned	AutoInc	Default
id	INTEGER	11	0	☑	☐	☑	Null
date_sistema	DATETIME	0	0	☐	☐	☐	Null
date_instrument	DATETIME	0	0	☐	☐	☐	Null
temperatura	FLOAT	0	0	☐	☐	☐	Null
conductividat	FLOAT	0	0	☐	☐	☐	Null
pressio	FLOAT	0	0	☐	☐	☐	Null
salinitat	FLOAT	0	0	☐	☐	☐	Null
velocitat_so	FLOAT	0	0	☐	☐	☐	Null

Figura 3.9: Lista de campos de una tabla

La pestaña "DLL", situada en la parte superior, como podemos ver en la figura anterior, nos ofrece la posibilidad de ver la creación de la tabla y de los campos en lenguaje SQL. Pero la pestaña más importante, porque es la que más utilizaremos, es "Data", a través de la cual se accede a toda la lista de registros contenidos en nuestra tabla donde podremos modificarlos. Desde aquí también se podrán insertar nuevos registros o eliminar los ya existentes de manera muy sencilla.

Properties | Fields | Indices | Triggers | Data | Dependencies | DDL

Find: 1000

Drag a column header here to group by that column

id	date_sistema	date_instrument	temperatura	conductividat	pressio	salinitat	velocitat_so
3.973.446	22/03/2012 9:51:11	22/03/2012 9:51:11	15,1629	0,11034	0,185	0,6845	1467,34
3.973.447	22/03/2012 9:51:21	22/03/2012 9:51:21	15,1624	0,11034	0,184	0,6845	1467,34
3.973.448	22/03/2012 9:51:31	22/03/2012 9:51:31	15,1614	0,11034	0,185	0,6845	1467,33
3.973.449	22/03/2012 9:51:41	22/03/2012 9:51:41	15,1605	0,11035	0,184	0,6846	1467,33
3.973.450	22/03/2012 9:51:51	22/03/2012 9:51:51	15,1591	0,11035	0,183	0,6846	1467,33
3.973.451	22/03/2012 9:52:01	22/03/2012 9:52:01	15,1581	0,11034	0,184	0,6846	1467,32
3.973.452	22/03/2012 9:52:11	22/03/2012 9:52:11	15,1579	0,11034	0,185	0,6845	1467,32
3.973.453	22/03/2012 9:52:21	22/03/2012 9:52:21	15,1573	0,11032	0,185	0,6845	1467,32
3.973.454	22/03/2012 9:52:31	22/03/2012 9:52:31	15,1573	0,11031	0,184	0,6844	1467,32
3.973.455	22/03/2012 9:52:41	22/03/2012 9:52:41	15,1566	0,1103	0,184	0,6843	1467,32
3.973.456	22/03/2012 9:52:51	22/03/2012 9:52:51	15,1572	0,11029	0,185	0,6842	1467,32
3.973.457	22/03/2012 9:53:01	22/03/2012 9:53:01	15,157	0,1103	0,183	0,6843	1467,32
3.973.458	22/03/2012 9:53:11	22/03/2012 9:53:11	15,1569	0,11032	0,184	0,6844	1467,32

Figura 3.10: Registros de una tabla

4. Análisis y Especificación

En el siguiente apartado se realiza un sencillo análisis para especificar los requerimientos que queremos que tenga la aplicación a desarrollar mediante el uso del **lenguaje UML**.

4.1 ¿Qué es el lenguaje UML?

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas software más conocido y utilizado actualmente. Estandarizado por el OMG (Object Management Group), se utiliza para especificar, visualizar y documentar los modelos de los sistemas de Software, incluyendo su estructura y diseño de manera que cumplan todos sus requerimientos. Es un lenguaje útil para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

UML es un lenguaje de modelado, no de programación, con el propósito general del modelado orientado a objetos, su uso es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha estado diseñado para poder modelar cualquier tipo de proyecto.

4.2 Análisis de la situación

Se quiere desarrollar una aplicación capaz de generar archivos NetCDF. Cada uno de estos archivos debe contener la información que se requiera en cada momento procedente del observatorio submarino OBSEA.

El observatorio dispone de dos Instrumentos CTD's, instrumentos utilizados para medir características del agua tales como temperatura, salinidad, presión, profundidad, etc… en zonas determinadas. De estos dos instrumentos será la procedencia de los datos que manipularemos. Ambos CTD's, en nuestro caso, emitirán datos de temperatura, conductividad, presión, salinidad y velocidad del sonido del mar.

Así pues, al disponer de dos fuentes de información, el usuario deberá tener la opción de elegir la procedencia de los datos, en este caso poder escoger el instrumento CTD deseado. Al igual que con la fuente de información, también queremos tener la opción de escoger, para cada uno de los archivos NetCDF generados, qué variables queremos incluir en ellos así como el intervalo de tiempo para el cual queremos dichos datos.

Una vez especificados instrumento, variables e intervalo de tiempo, la aplicación generará un archivo NetCDF con todos los datos relacionados con los parámetros elegidos previamente. Además de estos datos, el archivo NetCDF se complementará con otros datos adicionales como por ejemplo las unidades de cada una de las variables, información del instrumento escogido o información general sobre el centro de procedencia de dichos datos.

Figura 4.1: Instrumento CTD SBE 37-SMP

4.3 Especificación

4.3.1 Glosario

Con este glosario se pretende dar una definición clara, concisa y que no dé pie a ningún tipo de ambigüedad, de los principales términos utilizados en el análisis.

Archivo NetCDF: Archivo generado por la aplicación y que contiene toda la información que el usuario ha especificado.

Usuario: Miembro del grupo SARTI que utiliza la aplicación.

Observatorio: Construcción o lugar donde se observan fenómenos celestes, terrestres o submarinos. Estos se instalan en lugares que posean las condiciones apropiadas para la observación de aquello que se pretende estudiar. En este caso no referimos especialmente a un observatorio submarino. Un observatorio puede tener diferentes instrumentos.

Instrumento: Nos referimos a Instrumento como un equipo de medida que se conecta al observatorio para realizar las medidas deseadas. Un instrumento está compuesto por diferentes sensores que miden diferentes parámetros.

Variable: Cada uno de los diferentes tipos de datos procedentes de un Instrumento.

Variables del Instrumento: Lista de variables disponibles en un instrumento.

Variables NetCDF: Lista de variables escogidas para incluir en nuestro archivo NetCDF.

Intervalo de tiempo: Periodo de tiempo transcurrido entre dos fechas.

Fecha Inicial: Instante inicial de un intervalo de tiempo.

Fecha Final: Instante en el que finaliza el intervalo de tiempo.

4.3.2 Modelo Conceptual

Cabe mencionar, que el centro de desarrollo SARTI, ya dispone de un sistema para dar de alta nuevos observatorios e instrumentos. Por este motivo, daremos por hecho que en todo momento existirá como mínimo un observatorio en nuestra base de datos, en este caso OBSEA que ya fue dado de alta en su momento mediante un gestor, y que este contiene algún instrumento encargado de proporcionarnos datos.

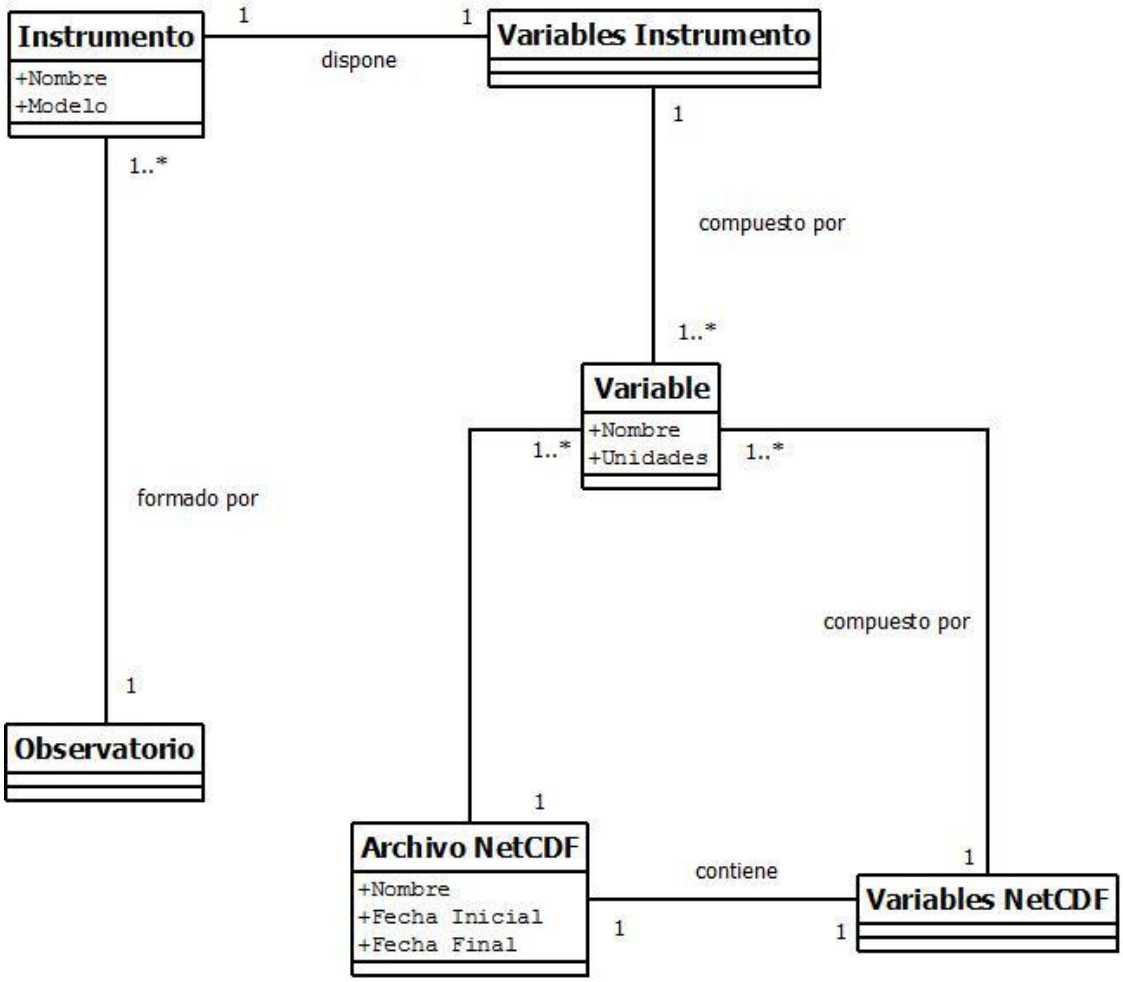


Figura 4.2: Modelo conceptual del análisis

Por eso, no contemplaremos estos procesos en nuestro caso de uso ya que forma parte de un proceso a parte totalmente transparente para el usuario que utilizara nuestra herramienta y que solamente se tendrá que preocupar del procedimiento de generación de los archivos NetCDF.

4.3.3 Lista de requerimientos

En la siguiente lista se detalla claramente las funciones que tendrá el sistema para cubrir el procedimiento de creación de archivos NetCDF.

R 1.1. Obtener un listado de Instrumentos.

R 1.2. Obtener un listado de las variables del instrumento.

R 1.3. Añadir las variables a nuestro archivo NetCDF.

R 1.4. Asignar un nombre al archivo NetCDF.

R 1.5. Asignar un intervalo de tiempo al archivo NetCDF.

R 1.6. Generar el archivo NetCDF.

4.3.4 Casos de uso

4.3.4.1 Especificación del caso de uso: Generar archivo NetCDF

Caso de uso: Generar archivo NetCDF

Actores: Usuario

Descripción: Se quiere generar un nuevo archivo NetCDF. Se selecciona el instrumento de procedencia de los datos, las variables a incluir en nuestro nuevo archivo, un intervalo de tiempo y se le asigna un nombre. Al hacer clic en el botón "Generar" se creará el archivo NetCDF. El sistema mostrará un mensaje indicándonos que el proceso se ha realizado correctamente. De lo contrario, también nos indicará si se ha producido algún error.

Requerimientos: R 1.1, R 1.2, R 1.3, R 1.4, R 1.5, R 1.6.

Transcurso normal de los acontecimientos:

Acciones de los Actores	Respuesta del Sistema
	1. El sistema nos muestra una lista desplegable con los instrumentos disponibles.
2. El usuario selecciona un instrumento.	
	3. El sistema nos indicará para qué periodo de tiempo disponemos de datos sobre el instrumento seleccionado.
	4. El sistema busca las variables relacionadas con el instrumento seleccionado y las muestra en la lista "Variables del Instrumento".
5. El usuario añade las variables deseadas en la lista "Variables NetCDF" para incluirlas en el archivo que generaremos.	
6. El usuario asigna un nombre al archivo.	
7. El usuario asigna una fecha inicial igual o superior a la indicada por el sistema.	

8. El usuario asigna una fecha final inferior o igual a la indicada por el sistema. 9. El usuario clica en el botón "Generar" para crear el nuevo archivo NetCDF.	10. El sistema nos muestra un mensaje conforme el archivo se ha creado correctamente.

i. Transcurso alternativo de los acontecimientos:

10. El usuario no ha añadido ninguna variable en la lista de "Variables NetCDF" y el sistema nos muestra un mensaje de que no puede crear el archivo sin ninguna variable añadida en la lista. Deberemos volver a realizar el paso 5.

ii. Transcurso alternativo de los acontecimientos:

10. El usuario ha añadido una Fecha Inicial incorrecta, el sistema muestra un mensaje de que no puede crear el archivo NetCDF para la fecha asignada. Debemos modificar la fecha introducida en el paso 7.

iii. Transcurso alternativo de los acontecimientos:

10. El usuario ha añadido una Fecha Final incorrecta, el sistema muestra un mensaje de que no puede crear el archivo NetCDF para la fecha asignada. Debemos modificar la fecha introducida en el paso 8.

4.4 Interfaz de la aplicación

En este apartado veremos los diferentes elementos que contendrá la interfaz de la aplicación que deberán cumplir con todos los requerimientos mencionados en los apartados anteriores.

En primer lugar necesitamos un listado, tal y como se muestra en la *figura 4.3*, que nos muestre los instrumentos disponibles en el sistema para poder escoger el que necesitaremos en cada momento.

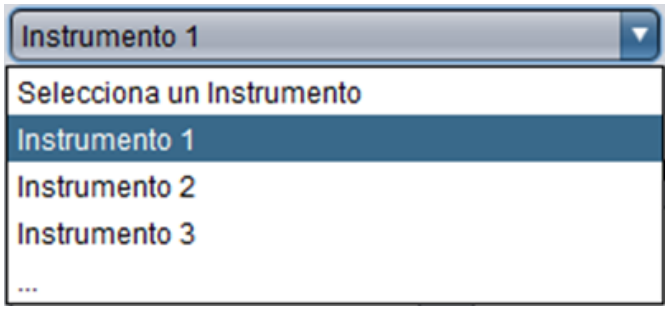


Figura 4.3: Lista de Instrumentos

Una vez seleccionado el instrumento requerido, en este caso uno de los dos CTD's que contiene el observatorio OBSEA, se nos mostrarán, en la lista "Variables Instrumento", las variables de las que hay disponibilidad de datos.

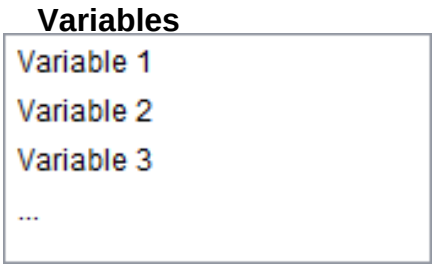


Figura 4.4: Lista de Variables disponibles

En este caso, ambos instrumentos nos mostrarán las mismas variables, ya que se trata de dos CTD's que miden las mismas características del agua.

El siguiente elemento que necesitaremos en nuestra aplicación será el listado de variables que habremos seleccionado para incluir en el nuevo archivo que generaremos. Para ello requeriremos de un botón "Añadir", al que clicaremos una vez hayamos seleccionado la variable que queremos añadir a la lista de "Variables NetCDF".

En este caso, como ejemplo, hemos querido añadir las variables “Temperatura” y “Presión”.

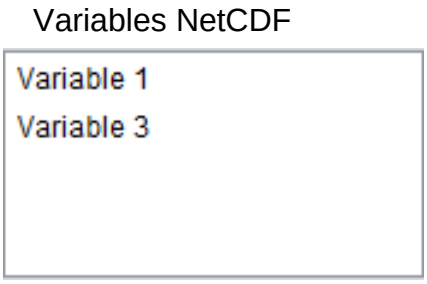


Figura 4.5: Lista de Variables añadidas

También dispondremos de varios campos donde el usuario deberá introducir un nombre para el archivo que generará y una fecha inicial y fecha final para determinar durante qué periodo de tiempo desea obtener los datos de las variables seleccionadas en el listado de “Variables NetCDF”.

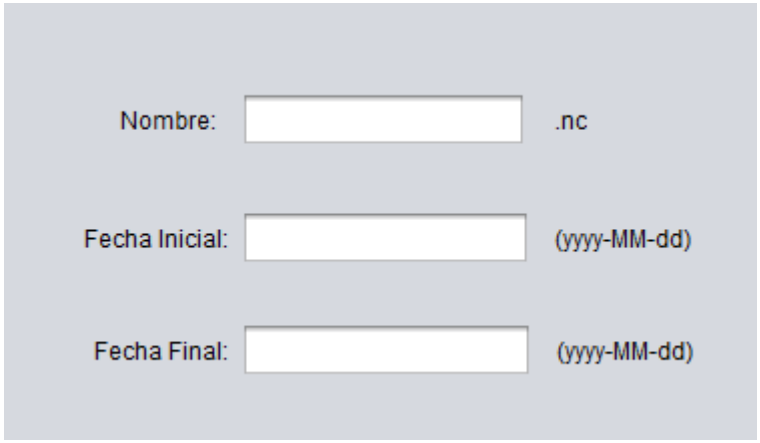


Figura 4.6: Campos de la aplicación

Como podemos ver en la *figura 4.7*, en el primer campo de texto es donde se le asignará el nombre a nuestro archivo, sin necesidad de especificar la extensión de los archivos NetCDF (.nc), ya que la aplicación se encargará de añadirla automáticamente. El segundo y tercer campo de texto será donde se le especifique una fecha inicial y una fecha final, sucesivamente, con el formato de fecha que se determina en la parte derecha de cada uno de los campos de texto (yyyy-MM-dd).

Por último, y no por ello menos importante, el elemento que nos falta es el botón "Generar", que una vez hayamos seleccionado las variables y rellenado cada uno de los campos mencionados con anterioridad, será el encargado de generar nuestro archivo NetCDF.

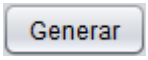


Figura 4.7: Botón "Generar"

Para finalizar, colocaremos cada uno de los elementos enseñados anteriormente en un contenedor para poder ver cuál es el diseño resultante de nuestra herramienta.

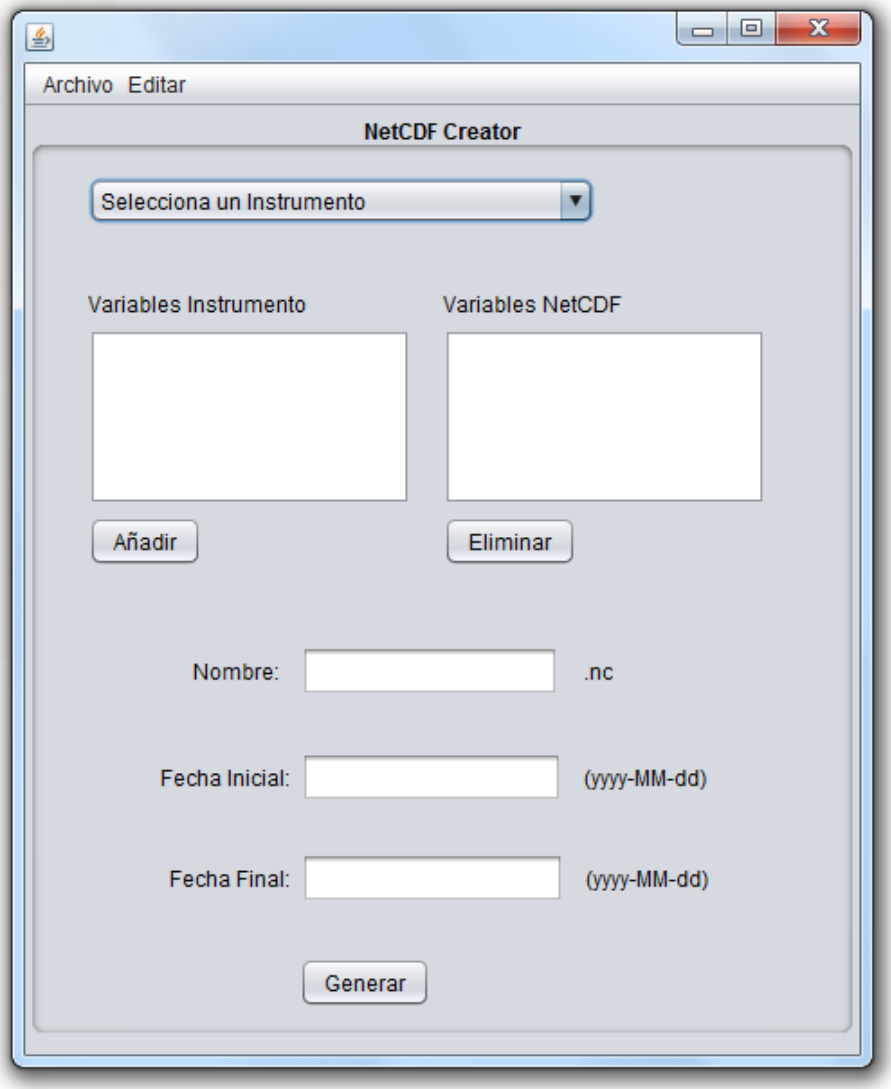


Figura 4.8: Interfaz general de la aplicación

5. Diseño

En el siguiente capítulo se explicará cual ha sido el criterio que se ha seguido a la hora de crear nuevas tablas o modificar algunas ya existentes para realizar el diseño de nuestra aplicación.

El objetivo en este apartado, es que nuestra aplicación software sea lo más "inteligente" posible, es decir, automatizar muchos de los procesos para que estos sean totalmente ajenos al usuario de la aplicación. Un ejemplo de este tipo de procesos automáticos sería que cuando el usuario decide añadir alguna de las variables al archivo NetCDF, la aplicación por defecto, al saber la procedencia de dicha variable, le asignaría una serie de atributos como podrían ser sus unidades, nombre estándar, etc…. De este modo simplificamos la faena al usuario. Lo mismo se produciría en la selección del instrumento y automáticamente, en el archivo, se añadiría información sobre este.

Para conseguir este propósito, a continuación veremos con detalle qué nuevas tablas hemos necesitado crear o qué modificaciones hemos hecho en algunas tablas que ya se encontraban en la base de datos del grupo SARTI y que hemos podido aprovechar para nuestro propósito.

En el diseño van a intervenir las siguientes 4 tablas:

- **obsea_ctd01**
- **obsea_ctd02**
- **obsea_instrument**
- **cf-1.6**

Las tres primeras, son tablas ya existentes en la base de datos de SARTI. Las dos primeras ("**obsea_ctd01**" y "**obsea_ctd02**") es donde se almacenan todos los datos de los dos instrumentos CTD's y no sufrirán ninguna modificación. Las utilizaremos para obtener los datos que introduciremos en los archivos NetCDF que vayamos creando.

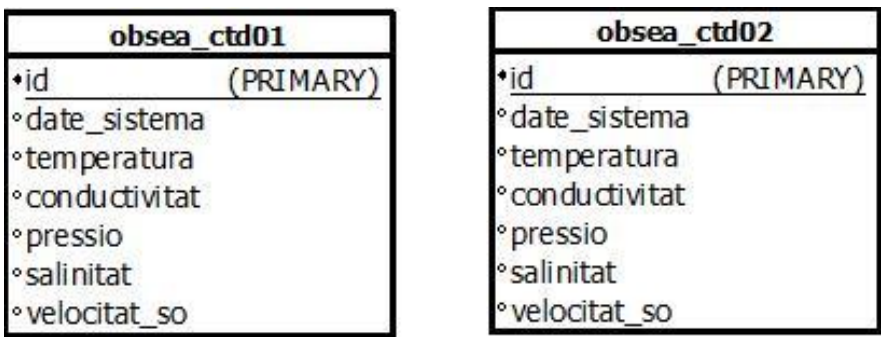


Figura 5.1: Diseño de las tablas obsea_ctd01 y obsea_ctd02

Este será el aspecto que tendrán ambas tablas:

id	date_sistema	date_instrument	temperatura	conductividat	pressio	salinitat	velocitat_so
259	04/03/2010 12:40:47	20100304114043	12,3808	4,3451	20,306	37,9975	1502,03
260	04/03/2010 12:40:57	20100304114053	12,381	4,34508	20,299	37,9972	1502,03
261	04/03/2010 12:40:57	20100304114053	12,381	4,34508	20,299	37,9972	1502,03
262	04/03/2010 12:41:07	20100304114103	12,3811	4,34508	20,305	37,9971	1502,03
263	04/03/2010 12:41:07	20100304114103	12,3811	4,34508	20,305	37,9971	1502,03
264	04/03/2010 12:41:17	20100304114113	12,3808	4,34505	20,419	37,9969	1502,03
265	04/03/2010 12:41:17	20100304114113	12,3808	4,34505	20,419	37,9969	1502,03

Figura 5.2: Visualización de los datos en la tabla obsea_ctd02

5.1 Tabla de los Instrumentos

En la tabla "**obsea_instrument**", ya existente, es donde se almacena información de los Instrumentos. Le podremos sacar provecho ya que es la encargada de relacionar el instrumento con la tabla a la que van a parar todos los datos que este genera. Por este motivo, no necesitaremos crear una nueva tabla para esta tarea, ya que esta la cumple a la perfección. Eso sí, además, de esta tabla necesitaremos obtener toda la información posible de los instrumentos del observatorio OBSEA, en este caso de ambos CTD's. Por este motivo, necesitamos añadir algunos de los campos ausentes que nos proporcionarán dicha información.

Los dos campos que añadiremos en la tabla "obsea_instrument" serán:

- **tipo_instrumento:** Contendrá la información del tipo de instrumento que nos está proporcionando los datos, como por ejemplo podría ser una cámara, hidrófono o CTD. En nuestro caso, como es obvio, se tratará de los CTD's.
- **modelo:** nos proporcionará información, todavía más exacta, del instrumento del que proceden los datos que obtenemos. Por ejemplo, "SBE 16Plus V2 SEACAT", modelo de uno de los CTD's.

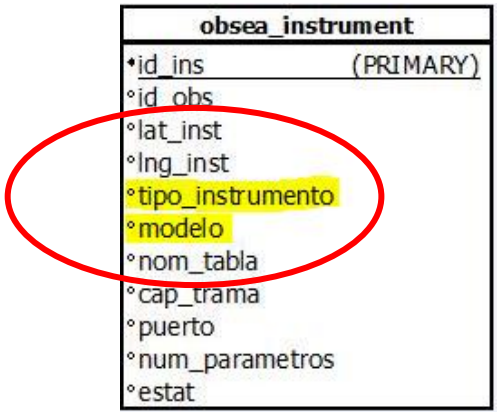


Figura 5.3: Diseño de la tabla obsea_instrument

Los cinco atributos incluidos en el círculo rojo que podemos ver en la *figura 5.1***,** son los campos que utilizaremos de esta tabla. Los atributos "lat_inst" y "lng_inst" nos indican la latitud y longitud en la que se encuentra el instrumento, información que será imprescindible incluir en nuestros archivos para lograr establecer su ubicación.

Los dos siguientes atributos, subrayados en amarillo, son los dos nuevos campos que le hemos añadido a esta tabla y que ya hemos explicado, anteriormente, cuál iba a ser su función. Por último, "nom_tabla", es el campo que contiene el nombre de la tabla destino en la que se alojan todos los datos del producido por el instrumento.

Una vez vistos todos los campos que utilizaremos de la tabla y especificada cuál será su función, veamos cuál será el aspecto resultante de la tabla "obsea_instrument":

id_inst	id_obs	lat_inst	lng_inst	tipo_instrumen	modelo	nom_tabla
1	1	41.122	1.12	CTD	SBE 16Plus V2	obsea_ctd02
6	1	41.122	1.12	CTD	SBE 37-SMP	obsea_ctd01

Figura 5.4: Visualización de los datos en la tabla obsea_instrument

5.2 Tabla CF-1.6

Esta será la tabla con más protagonismo del proyecto. En ella se almacenará toda la información relacionada con el vocabulario estándar, según la "Climate and Forecast Conventions", que deberemos utilizar para identificar cada una de las variables, juntamente con sus atributos, en los archivos NetCDF de manera que cualquier persona que quiera verlos sea capaz de entender qué tipo de información se almacena en él.

El objetivo principal de esta tabla es "traducir" del vocabulario utilizado por nosotros al vocabulario estandarizado y común para todo el mundo. Por ejemplo, cuando nosotros nos referimos a "temperatura", estamos haciendo referencia a la temperatura, en este caso del agua, ya que tenemos conocimiento de que estos datos provienen de un observatorio submarino y ese es el nombre que nosotros le hemos adjudicado. Si publicáramos una serie de datos, atribuyéndole como nomenclatura dicho nombre (temperatura), en el momento que un sueco o un canadiense quisieran acceder y visualizar dicha información, no solamente no sabrían qué quiere decir la palabra "temperatura", sino, que en el mejor de los casos y con un poco de suerte, consiguiendo averiguar su significado, tampoco sabrían a qué tipo de temperatura estamos haciendo referencia (temperatura del aire, temperatura del agua,…) y podrían llegar a una conclusión totalmente errónea. Además, utilizando nuestro propio vocabulario, el archivo será totalmente interpretable por parte de los visualizadores estandarizados. Es aquí donde la creación de una convención, para este tipo de archivos, entra en juego y nos proporciona un extenso documento donde especifica detalladamente el nombre a

asignar a cada una de las variables en cada situación. En el ejemplo propuesto anteriormente, al estar hablando de la temperatura del agua, el nombre correcto según la convención sería "sea_water_temperature".

A continuación, se explicará qué campos deberá contener la nueva tabla y la función que tendrá cada uno de ellos:

- **nombre:** Este campo contendrá el nombre que nosotros asignamos a las variables en nuestro vocabulario natural, por ejemplo "temperatura". Será utilizado cuando, al seleccionar un instrumento, nos aparezca el listado de sus variables. De este modo, cualquier usuario de la aplicación conocerá su significado y sabrá a qué tipo de dato hace referencia la variable.

- **pv_name:** En este campo encontraremos el nombre que se le asignará a la ProtVariable cuando la declaremos. Por ejemplo "temperature".

- **standard_name:** El nombre estándar, especificado por la convención (CF-1.6), de cada una de las variables que utilizaremos. Siguiendo el ejemplo anterior de la temperatura, a esta variable deberemos llamarla "sea_water_temperature".

- **long_name:** "long_name" es un atributo que sirve para complementar y dar más información sobre las ProtoVariables. Por eso, este campo contendrá un nombre, no estándar, de la ProtoVariable. Aunque no esté definido exactamente vocabulario exacto que se debe utilizar en una variable "long_name", sí que tiene que cumplir algunas normas establecidas por la convención. Debe ser un nombre inteligible para todo el mundo, por eso se tratará de un nombre en inglés, ya que es la lengua universal. Por ejemplo, a esta variable le podremos asignar el nombre "water temperatura".

- **unidades:** Se especificará en qué unidades está descrita la información de cada una de las ProtoVariables. La nomenclatura de las unidades también está especificada en la convención, así pues no nos servirá de nada nombrar con "ºC" a las unidades con la que estamos midiendo la temperatura, sino que no nos quedará más remedio que obedecer a la "CF Conventions" para que nuestros archivos sean correctamente interpretados. El nombre de las unidades de medida de la temperatura del agua deberá ser "Celsius".

- **tipo_dato:** El tipo de dato es un atributo obligatorio en la declaración de una ProtoVariable, así pues, este campo contendrá esta información. Un archivo NetCDF soporta los siguientes tipos de datos: ***boolean, byte, char, short, int, long, float, double*** y ***String.*** Por ese motivo, el tipo de datos deberá ser uno de los mencionados.

- **instrumento:** Este campo nos servirá para vincular esta tabla con la tabla "obsea_instrument".

Una vez descritos cada uno de los campos que formarán parte de la tabla "cf-1.6", pasemos a ver cuál es su diseño.

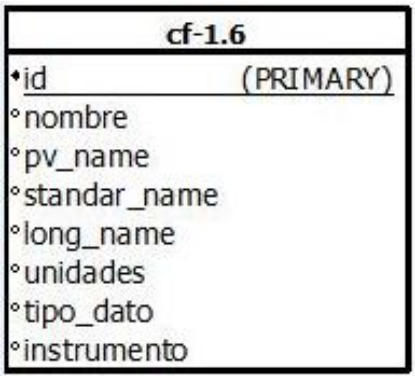


Figura 5.5: Diseño de la tabla cf-1.6

Y este será el aspecto que tendrá la tabla:

nombre	pv_name	standard_name	long_name	unidades	tipo_dato	instrumento
temperatura	temperature	sea_water_temperatur	water temperature	Celsius	double	obsea_ctd01
conductividat	conductivity	sea_water_electrical_cond	water conductivity	S m-1	double	obsea_ctd01
pressio	pressure	sea_water_pressure	water pressure	decibar	double	obsea_ctd01
salinitat	salinity	sea_water_salinity	water salinity	1e-3	double	obsea_ctd01
velocitat_so	water_speed	sea_water_speed	sea water speed	m s-1	double	obsea_ctd01

Figura 5.6: Visualización de los datos en la tabla cf-1.6

Veamos ahora un diagrama del diseño global de las tablas vistas con anterioridad:

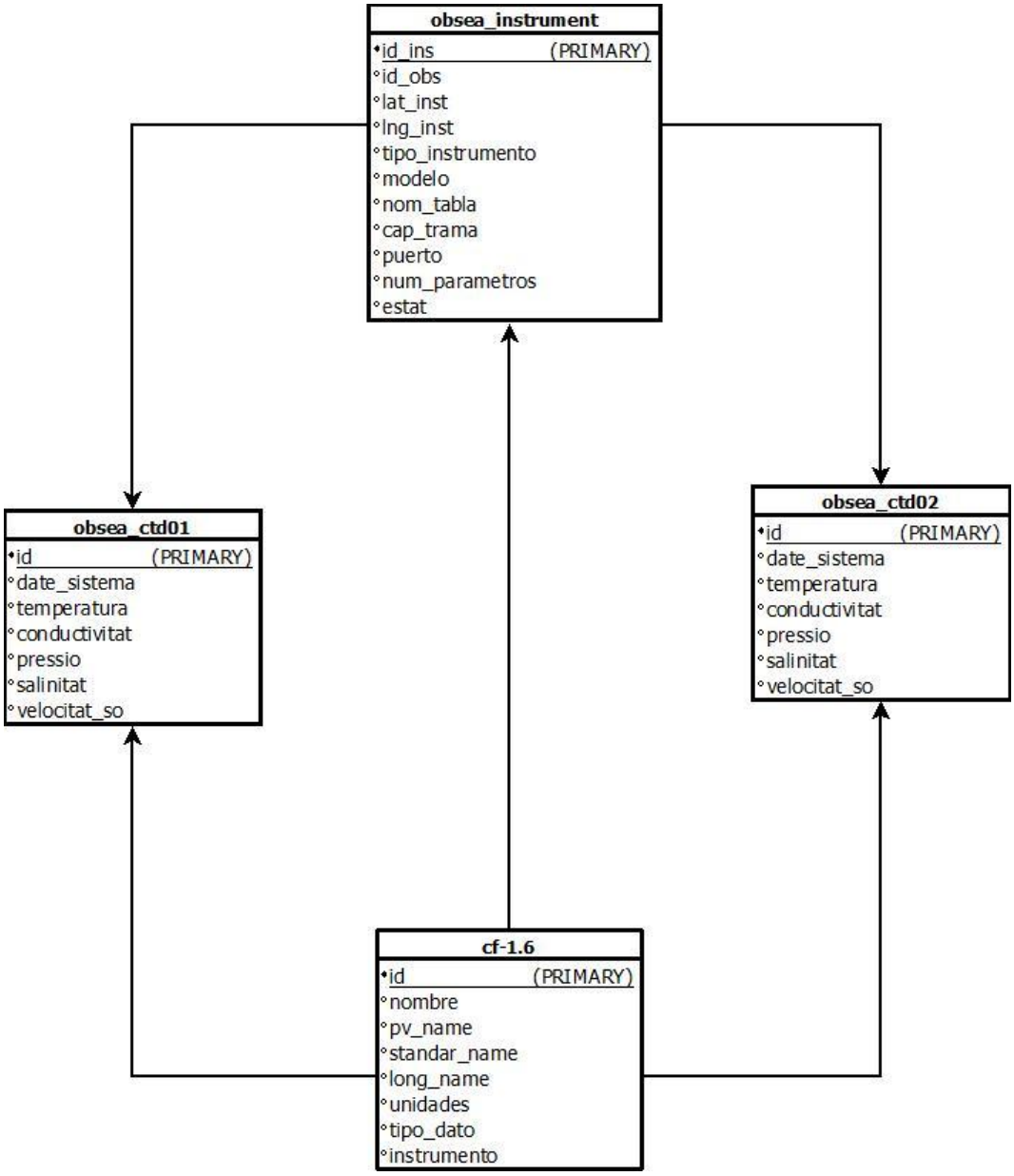


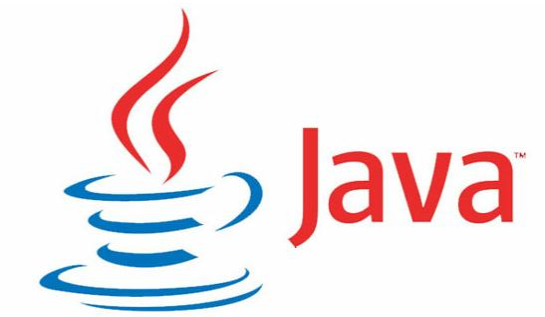
Figura 5.7: Diagrama UML general del diseño de la Base de Datos

6. Implementación

Una vez ya hemos hecho un análisis de todos los aspectos que debe cubrir la aplicación a desarrollar y de las tablas que necesitaremos según el diseño anterior, pasaremos a su implementación. Durante este capítulo se explicará detalladamente los aspectos contemplados, se mostrarán fragmentos de código importantes que hayamos utilizado, funciones específicas, se explicará qué librerías se han usado y para qué funcionalidad, y todo aquello utilizado durante el proceso de implementación que se crea conveniente explicar.

Para empezar, explicaremos algunos conceptos sobre el lenguaje de programación escogido para la implementación, Java.

6.1 Java: Lenguaje de Programación



Java es un lenguaje de programación "Orientado a Objetos" (OO), de una plataforma independiente.

El lenguaje de programación Java se creó con cinco objetivos principales:

1. Debe usar el paradigma de la programación orientada a objetos.
2. Debe permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debe incluir por defecto soporte para el trabajo en red.
4. Debe diseñarse para ejecutar código en sistemas remotos de una forma totalmente segura.
5. Debería ser fácil de utilizar y tomar lo mejor de otros lenguajes orientados a objetos, como por ejemplo C++.

Esta primera característica del lenguaje se refiere al método de programación y diseño del mismo. La intención de un lenguaje orientado a objetos es diseñar el software de forma que los diferentes tipos de datos estén unidos a sus operaciones. De este modo, las funciones y los métodos se combinan en entidades llamadas "Objetos".

Una manera de imaginarnos a un objeto, es como si fuese un paquete que contiene cuál va a ser su comportamiento (el código) y cuál va a ser su estado (datos). Uno de los principios fundamentales del lenguaje es separar todo aquello que cambia de las cosas que permanecen inalteradas. Esta separación en objetos nos ofrece una base más estable para el desarrollo y diseño de una aplicación software.

El objetivo de la programación orientada a objetos es la creación de entidades (en este caso los objetos) más genéricas que permitan su reutilización entre diferentes proyectos. En este sentido, un objeto podría verse como una pieza que puede ser reutilizada, posibilitando así a la industria del software en general, a construir proyectos utilizando componentes ya existentes y que su correcto funcionamiento ya ha sido probado, conduciendo a una reducción importante del tiempo del desarrollo de dichos proyectos.

Otra característica, la independencia de la plataforma, nos indica que programas escritos en lenguaje Java, pueden ejecutarse igualmente en cualquier tipo de hardware.

Por estos motivos, Java se ha convertido en un recurso prácticamente imprescindible en el mundo del software ya que permite a los desarrolladores realizar una gran variedad de tareas tales como:

- Desarrollar aplicaciones software en una plataforma y ejecutarlo en prácticamente cualquier dispositivo.
- Crear programas para que funcionen en navegadores y servicios Web.
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.

6.1.1 Herramientas de Java que utilizaremos

- **JDK (Java Development Kit):** Java Development Kit, es el kit de desarrollo oficial del lenguaje de programación Java, incluido en el NetBeans IDE, que utilizaremos. Este software nos provee de herramientas de desarrollo para la creación de programas en Java, así como una máquina virtual de Java, fundamental para ejecutar las clases de los programas que desarrollamos. Además de esta máquina virtual, cuenta con un amplio surtido de herramientas como por ejemplo el compilador "*javac*" o el depurador de posibles errores.

- **JDBC (Java DataBase Connectivity):** La tecnología JDBC es una interface de programación de aplicaciones (API) que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, por lo tanto, será el encargado de establecer la conexión entre nuestra aplicación y nuestra base de datos. Es totalmente independiente del sistema operativo donde se ejecute o del tipo de base de datos a la cual tenga que acceder.

  La API JDBC se nos presenta como una colección de interfaces Java y métodos de gestión que manejarán la conexión hacia cada modelo específico de base de datos que vayamos a utilizar.

  Para poder utilizarlo en nuestra aplicación, deberemos integrar la biblioteca de conexión apropiada al modelo de base de datos que utilicemos. A partir de este momento, podremos realizar cualquier tipo de tareas con la base de datos a las que tengamos acceso: consultas, actualizaciones, creación, modificación o borrado de campos y tablas.

  JBDC, también nos ofrece un paquete llamado "*java.sql*" que contiene clases implementadas muy útiles para trabajar con las bases de datos.

6.1.2 Bibliotecas Utilizadas

La mayoría de sistemas operativos de hoy en día, ofrecen código para simplificar las tareas de programación. Este código toma la forma de un conjunto de bibliotecas dinámicas que las aplicaciones utilizan cuando lo necesitan. Pero, la Plataforma Java está pensada para ser independiente de cualquier sistema operativo por lo que sus aplicaciones no pueden ayudarse de dichas funciones, las cuales son diferentes para cada uno de los sistemas operativos. Lo que hace la Plataforma Java, es ofrecer un conjunto de bibliotecas estandarizadas que contienen muchas de las funciones disponibles en los sistemas operativos.

A continuación nombraremos algunas de las bibliotecas que hemos utilizado a lo largo del proyecto, explicando cuál será su función.

6.1.2.1 NetCDF-Java Library (versión 4.2)

Las bibliotecas de Java tienen como propósito ofrecer al programador un conjunto definido de funciones para realizar tareas comunes. En nuestro caso, a lo largo del proyecto, además de las bibliotecas más comunes de Java utilizaremos las **bibliotecas NetCDF**, uno de los principales motivos por el cual he escogido este lenguaje de programación para el desarrollo de mi aplicación, ya que se trata de una gran variedad

de funciones específicas, proporcionadas por UNIDATA, porque Java no las incluye por defecto, para tratar con todo lo relativo a los archivos NetCDF.

6.1.2.2 JBDC - MySQL Connector Java (versión 5.1.19)

Como hemos podido ver anteriormente, JBDC es el encargado de establecer la conexión entre la aplicación y la base de datos. En este caso, al estar trabajando con una base de datos MySQL, éste deberá ser el conector java que deberemos utilizar.

6.1.2.3 Biblioteca gráfica javax.swing

Swing es una biblioteca gráfica para Java. Incluye todo tipo de objetos para diseñar la interfaz gráfica tales como paneles, etiquetas, cajas de texto, menús desplegables, tablas, botones, etc.

Desde los inicios de Java, este ya contaba con una biblioteca de componentes gráficos llamada AWT. Esta biblioteca fue creada como una API estandarizada que permitía utilizar los diferentes componentes de cada sistema operativo. Por ejemplo, una aplicación funcionando en Microsoft Windows utilizaría el botón estándar de Windows, en cambio, una aplicación corriendo en un sistema operativo UNIX usaría el botón estándar de Motif[5]. Pero esta tecnología no funcionó por dos motivos:

- Al depender fuertemente de los componentes del Sistema Operativo, el programador que utiliza AWT solamente dispone de las funcionalidades comunes en todos los Sistemas Operativos.
- El comportamiento de los controles es muy diferente entro los diferentes Sistemas Operativos, lo que hace muy difícil construir aplicaciones portables.

Swing introdujo un mecanismo que permitía que el aspecto de cada componente de una aplicación pudiese cambiar sin introducir grandes cambios en el código de la aplicación. Al añadirse este mecanismo, Swing permitió emular la apariencia de los componentes de cada Sistema Operativo manteniendo las ventajas de la independencia de la plataforma.

[5] Motif: Biblioteca para la creación de entornos gráficos en sistemas Unix.

6.1.2.4 Otras bibliotecas

Además de las bibliotecas mencionadas anteriormente utilizaremos algunas de las más comunes que incluye nuestro entorno de programación Java:

- Biblioteca **java.text**: Proporciona clases e interfaces para la manipulación y el formato de textos, fechas, número y mensajes.
- Biblioteca **java.util**: Contiene una colección de clases útiles. Entre estas clases se encuentras muchas para manipular datos genéricos (Diccionarios, Vectores, Tablas de Hash).
- Biblioteca **java.io**: Proporciona la entrada y salida del sistema a través de flujos de datos, serialización y el sistema de archivos.

6.1.3 ¿Cómo podemos utilizar estas bibliotecas?

En el caso de las bibliotecas "javax.swing", "java.text", "java.util" y "java.io", al ser bibliotecas que ya vienen incluidas en nuestro IDE, para utilizarlas solamente tendremos que añadir una pequeña línea de código en las clases que requieran de su utilización. En la parte superior de nuestra clase, al principio del todo de nuestro código, deberemos importar dichas librerías como vamos a ver a continuación:

```
import java.util.*;
import java.text.*;
import java.io.IOException;
import javax.swing.DefaultListModel;
import javax.swing.JOptionPane;
```

En las dos primeras importaciones, podemos ver que estas van seguidas de un "*", esto significa que estamos llamando a todas las clases de las bibliotecas java.util y java.text. Contrariamente, en el resto de importaciones, solamente, estamos llamando a la clase especificada. Por ejemplo, en las importaciones de la biblioteca swing, solamente estamos requiriendo las clases "DefaultListModel" y "JOptionPane".

El caso de las dos primeras bibliotecas mencionadas, "NetCDF-Java Library" y "MySQL Connector Java", es distinto, ya que no disponemos de ellas en nuestro entorno de desarrollo. Por eso, deberemos buscarlas y descargarlas.

Ambas bibliotecas pueden ser descargadas desde sus respectivos sitios web oficiales, www.unidata.ucar.edu y www.mysql.com. De este modo, una vez descargadas en nuestro equipo lo que obtendremos serán dos archivos, con extensión ".jar", que contendrán todas las clases y funciones de dichas bibliotecas.

Para integrarlas en nuestro proyecto y de este modo, poder hacer uso de ellas, deberemos añadirlas al listado de bibliotecas que usará el proyecto determinado. El proceso es muy simple, solamente tendremos que ir al menú "*Propieties*" del proyecto y seleccionar el apartado "*Libraries*". Allí, nos aparecerá un listado (en blanco si todavía no hemos añadido ninguna) y un explorador donde buscar la ruta donde tenemos descargadas las librerías para finalmente añadirlas.

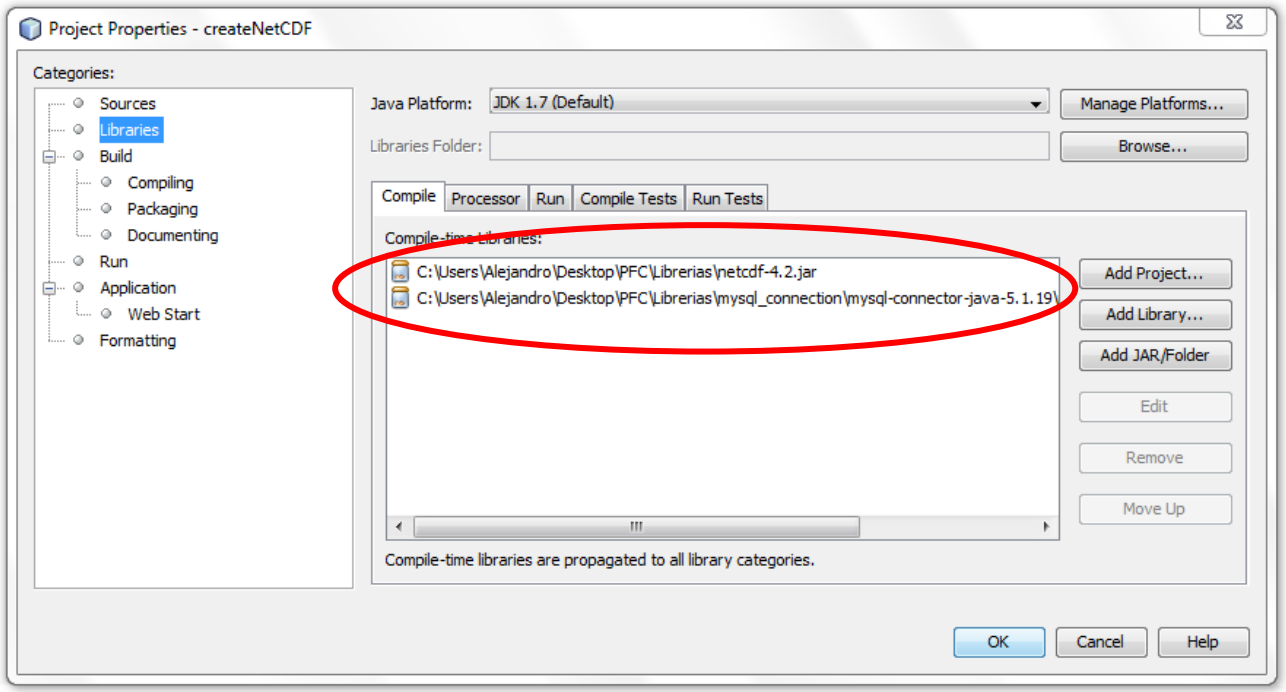


Figura 6.1: Menú "Libraries" de NetBeans IDE

Una vez hayamos acabado de añadir las bibliotecas al proyecto, al igual que con el resto de bibliotecas anteriores, solamente deberemos hacer un "*import*" de ellas, en las clases que requieran de sus funciones.

```
import ucar.netcdf.*;
import java.sql.*;
```

6.2 Desarrollo de la aplicación

6.2.1 Declaración y construcción de variables

En el siguiente apartado veremos, detalladamente, parte del código utilizado en la implementación, así como alguna de las funciones específicas de cada una de las librerías mencionadas en la sección anterior, sobretodo de la *NetCDF-Java Library*, que será, con diferencia, la más utilizada, ya que es la que nos proporcionará la gran mayoría de funciones relacionadas con los archivos NetCDF que deberemos crear.

El primer paso y uno de los más importantes será la declaración de nuestra dimensión. Recordamos que la dimensión es la variable que nuestros datos tomarán como referencia y sobre la que irán adquiriendo diferente valor. Las dimensiones más comunes son el tiempo, latitud, longitud y profundidad. Como en este caso queremos producir datos de un observatorio submarino estático, no nos hará falta contemplar ni latitud, ni longitud, ni profundidad, ya que su posición, en ningún momento cambiará.

Entonces, en este caso, solamente será necesario declarar una dimensión. El tiempo. Lo haremos utilizando el siguiente código:

```
Dimension timeD = new UnlimitedDimension("time");
```

Con esta línea lo que conseguimos es obtener una variable "timeD" de tipo "Dimension". Como crearemos archivos para diferentes intervalos de tiempo: un día, una semana, un mes, un año, etc… el tamaño de esta variable lo declararemos como ilimitado (UnlimitedDimension), para que se adapte a cualquiera de los casos.

Una vez tenemos declarada la dimensión, debemos añadirle todos los atributos que sean necesarios para su posible interpretación por parte de los visualizadores de datos.

```
String timePName = "time";
Class timePType = double.class;  // o Double.TYPE
Dimension[] timePDims = {timeD};
Attribute timePLongName = new Attribute("long_name", "time");
Attribute timePStandardName = new Attribute("standard_name", "time");
Attribute timePCalendar = new Attribute("calendar", "gregorian");
Attribute timePUnits = new Attribute("units","seconds since 1970-01-01 00:00");
Attribute timePZone = new Attribute("time_zone", "CET");
Attribute[] timePAtts = {timePLongName, timePStandardName,
                         timePCalendar, timePZone, timePUnits};

ProtoVariable timeP = new ProtoVariable(timePName, // nombre
                                        timePType, // tipo de dato
                                        timePDims, // dimension
                                        timePAtts  // atributos
                                       );
```

En el fragmento de código anterior podemos ver todos los atributos que se le asignan a la variable "time", en este caso, la dimensión del archivo a generar. Especificamos el tipo de dato que contendrá, le asignaremos un "*standard_name*" y un "*long_name*". Al tratarse de una variable de tiempo, en este caso también le especificamos un tipo de calendario o una zona horaria, aunque estos datos son complementarios. Una vez hayamos acabado de declarar nuevos atributos, los introducimos todos ellos en un "array" llamado "timePAtts". Posteriormente, declararemos una nueva *ProtoVariable* y añadiremos todos los atributos, que hayamos creado antes, a esta nueva variable juntamente con el tipo de dato y la dimensión (en este caso ella misma).

Cabe mencionar, que aunque "time" sea nuestra "dimension" también tiene la misma estructura que el resto de las variables, en el caso de los archivos NetCDF llamadas *ProtoVariables*. Por eso, también debemos asignarle atributos, como hemos visto con el código anterior.

El siguiente paso es declarar el resto de variables. Como veremos en el siguiente código, el procedimiento será exactamente el mismo que con la variable anterior.

```
String pPName = "pressure";
Class pPType = double.class;
Dimension[] pPDims ={timeD};
Attribute pPLongName = new Attribute("long_name", "water pressure");
Attribute pPStandardName = new Attribute("standard_name", "sea_water_pressure");
Attribute pPUnits = new Attribute("units", "decibar");
Attribute[] pPAtts = {pPLongName, pPStandardName, pPUnits};
pP = new ProtoVariable(pPName,pPType,pPDims,pPAtts);
```

Vemos que en la declaración de la ProtoVariable "*pressure*", la dimensión es "*timeD*", la única dimensión declarada para nuestros archivos y que será común para todas las demás variables.

6.2.2 Declaración de atributos globales

Hasta el momento, todos los atributos que hemos ido viendo a lo largo de la implementación formaban parte de una variable. Pues bien, ahora vamos a ver de qué forma podemos declarar atributos globales, es decir, atributos que determinen alguna característica del documento en general y que haga referencia a todo el contenido que hay dentro de él.

La forma de hacerlo es muy sencilla porque la estructura y la manera de hacer su declaración es exactamente la misma que hemos utilizado anteriormente para los atributos pertenecientes a las variables. Lo único que los distingue es que estos no

formarán parte de ningún "array" de atributos que posteriormente incluiremos en alguna ProtoVariable, sinó que serán totalmente independientes.

A continuación podemos ver la declaración de tres atributos globales:

```
Attribute titulo = new Attribute("title", "Datos Observatorio OBSEA");
Attribute netcdf_version = new Attribute("netcdf_version", "4.2");
Attribute Convention = new Attribute("Convention", "CF-1.6");
```

Como vemos, cada uno es un atributo independiente y la información que contienen hace referencia al archivo NetCDF completo, ya que tienen efecto sobre la totalidad de su contenido. He querido mostrar estos tres atributos, ya que son algunos de los más comunes que podemos encontrar en este tipo de archivos.

- **title**: Adjudica el título especificado a nuestro archivo.
- **netcdf_version**: Nos indica con qué versión de archivo estamos tratando. Esta información puede sernos de utilidad, ya que dependiendo la versión de librerías NetCDF que hayamos utilizado, el archivo podrá ser visualizado o no por determinados visualizadores dependiendo de su compatibilidad con las diferentes versiones NetCDF.
- **Convention**: Especifica qué versión de la "*Climate and Forecast Conventions*" hemos utilizado para identificar cada uno de los componentes del archivo.

Además de los atributos globales vistos, podemos añadir una gran variedad de ellos para dar la mayor cantidad posible de información al usuario que, en un futuro, consultará nuestra información.

Otros ejemplos de atributos globales podrían ser:

- **creator_name:** Nombre de la persona o centro que ha generado el archivo.
- **institution:** Institución a la que pertenecen los datos contenidos en el archivo.
- **CTD_Instrument:** Modelo del Instrumento CTD que ha generado la información.
- **date_created:** Fecha de creación del archivo.
- **date_update:** Fecha en la que se han actualizado los datos del archivo.
- **time_coverage_start:** Fecha inicial del periodo de tiempo de los datos que componen el archivo.

- **time_coverage_end:** Fecha final del periodo de tiempo de los datos que componen el archivo.
- **keywords:** Palabras clave que pueden ayudar a los usuarios a hacerse una idea del tipo de contenido que hay en el archivo NetCDF.

Todos ellos, útiles para facilitar la compresión de los datos contenidos en el archivo y saber más información sobre la procedencia de los mismos.

6.2.3 Estructura del archivo NetCDF

Una vez finalizada la fase de declaración y composición de cada una de las variables que formarán parte del archivo, llega el momento de establecer y determinar cuál será la estructura del mismo. Para ello, es necesario definir el esquema que adoptará el archivo NetCDF.

La declaración del esquema (*Schema*) se realizará de la siguiente manera:

```
Schema schema = new Schema(
    new ProtoVariable[]{timeP, latP, lonP, depthP, tP, cP, pP, sP, vP},
    new Attribute[] {titulo, netcdf_version, Convention}
);
```

Lo único que haremos será declarar una nueva variable de tipo “Schema” en la que añadiremos dos “arrays”, uno formado por todas las ProtoVariables que habremos declarado anteriormente y otro compuesto por todos los atributos globales que también hemos declarado con anterioridad. Así pues, de este modo todo el contenido del archivo quedará establecido dentro del esquema.

Una vez ya tenemos hecha la estructura del archivo, el siguiente paso será crear dicho archivo de la forma que se mostrará a continuación:

```
NetcdfFile nc = new NetcdfFile(fileName,
                               true, // sobreescribir un archivo ya existente
                               true, // rellenar por defecto
                               schema // plantilla de los metadatos
                              );
```

Con estas simples líneas, conseguimos crear un elemento "nc" de tipo NetcdfFile, que a lo largo de toda la implementación hará referencia al archivo NetCDF. Por parámetros especificamos un nombre, en este caso almacenado en una variable String "fileName". Por ejemplo:

```
String fileName  = "obsea_Data.nc";
```

Los dos siguientes parámetros nos servirán para, primeramente, decirle al programa que al generar un nuevo archivo NetCDF, con un nombre idéntico a alguno creado anteriormente, éste se sobrescriba, eliminando a su predecesor. En el segundo parámetro, le diremos si, en caso de tratarse de un archivo vacío, queremos rellenar sus variables con algún tipo de valor por defecto, como podrían ser "0" o "NULL".

Finalmente, le proporcionaremos un esquema, concretamente el que hemos determinado en el paso anterior.

6.2.4 Obtener los datos de la Base de Datos

Con todos los pasos realizados en el apartado anterior, hemos conseguido crear la estructura de un archivo NetCDF aunque este todavía no contiene ningún dato, está vacío. Por ello, a continuación se mostrará el proceso para conseguir introducir los datos que deseemos dentro de cada una de las variables a las que correspondan.

Los datos que queremos introducir en el archivo se encuentran alojados en las tablas de la base de datos del centro de desarrollo SARTI, concretamente en "obsea_ctd01" y obsea_ctd02, que hacen referencia a los datos proporcionados por los CTD's "*SBE 37-SMP MicroCAT"* y *"SBE 16Plus V2 SEACAT"* respectivamente. Por precaución y para no dañar las tablas mencionadas durante las pruebas realizadas durante la implementación de la aplicación, he creado en mi base de datos local dos tablas idénticas a ellas, con una muestra importante de datos, que simulan a las tablas del centro. Las tablas del centro son de un gran tamaño y contienen millones de registros en ellas, por ese motivo no las he copiado por completo en mi base de datos local. Aun así, se dispone de cientos de miles de registros, muestra totalmente suficiente, para realizar las pruebas necesarias a lo largo del desarrollo del proyecto.

Pues bien, una vez aclarado esto, pasamos a ver como se realiza la conexión a la base de datos local llamada "iso". Es aquí donde entrará en acción el paquete de "java.sql" que nos ofrece JDBC, la API de java añadida a nuestras bibliotecas en los pasos previos a la implementación.

El primer paso consiste en cargar el controlador JDBC, un objeto Driver, específico dependiendo del tipo de base de datos que utilizaremos, que define la manera en la que se ejecutan las instrucciones para esta base de datos en particular (MySQL).

El método más sencillo es utilizar el método *forName()* de la clase *Class*. Este método nos generará un objeto de la clase especificada:

```
Class.forName("com.mysql.jdbc.Driver");
```

También debemos determinar la URL donde se encuentra nuestra base de datos:

```
String url = "jdbc:mysql://localhost:3306/iso";
```

Una vez se ha determinado la URL, se podrá establecer una conexión a esta base de datos.

También necesitaremos un objeto *Connection*, es el principal objeto que utilizaremos para proporcionar un vínculo entre la base de datos y la aplicación en Java. Este objeto *Connection* nos proporcionará una conexión estática de nuestra base de datos. Esto quiere decir que hasta que no se llama al método *close()*, para cerrar la conexión o se destruya el objeto *Connection*, la conexión a la base de datos permanecerá activa.

La manera más usual de establecer la conexión a una base de datos es invocando al método *getConnection()* de la clase DriverManager. En muchas ocasiones, las bases de datos están protegidas con nombres de usuario ("*username*") y contraseña ("*password*") para restringir el acceso a ellas. El método *getConnection()* nos permite que tanto el nombre del usuario como la contraseña sean pasadas como parámetros, de igual forma que la `Connection myCon;` eriormente.

```
String url = "jdbc:mysql://localhost:3306/iso";
String username = "root";
String password = "";

myCon = DriverManager.getConnection(url,username,password);
```

Como podemos ver, en este caso, el campo "*password*" está vacío, ya que al tratarse de una base de datos local, nadie más excepto yo tendrá acceso y no he creído necesario establecerle una contraseña.

Del mismo modo que necesitamos un objeto *Connection* también necesitamos el objeto *Statement*, que será el que nos permitirá ejecutar sentencias SQL y enviarlas a la base de datos:

```
Statement myStmt;
myStmt = myCon.createStatement();
```

Y un objeto *result* de tipo *ResultSet*, que almacenará los resultados obtenidos en la consulta. Una vez tenemos declarados cada uno de los componente mencionados ya podemos hacer nuestra consulta SQL a la base de datos. En este caso, y de ejemplo, haremos una consulta en la que seleccionaremos tantos campos como ProtoVariables hemos declarado:

- **date_sistema**: Será de donde procederán todos los datos que introduciremos en la dimensión "time".
- **temperatura, conductivitat, pressio, salinitat, velocitat_so**: Serán los campos de donde obtendremos todos los datos para el resto de ProtoVariables que hemos declarado.

Veamos qué aspecto tendrá la consulta SQL:

```
ResultSet result;
String sql = "SELECT date_sistema, temperatura, conductivitat,"
        + "pressio, salinitat, velocitat_so FROM obsea_ctd01";

result = myStmt.executeQuery(sql);
```

El siguiente paso es almacenar todos los datos que obtenemos de la consulta. Para ello necesitamos algún tipo de objeto para poder ir almacenando los resultados. En este caso la opción adoptada ha sido la de almacenarlos en listas, ya que se trata de un objeto dinámico que permite modificar su tamaño según el número de datos que tengamos que introducir en ellas en cada momento.

Declararemos un objeto lista, para almacenar todos los datos del campo "date_instrument" de la base de datos, de la siguiente manera:

```
List <Double> dateList = new ArrayList<Double>();
```

Como anteriormente hemos especificado que los datos que almacenaríamos en la ProtoVariable "time" serían de tipo *Double*, declaramos una lista que pueda contener este tipo de datos. En este momento se nos plantea un problema porque el tipo de dato del campo "date_instrument" de la base de datos contiene datos tipo *Date*. Podríamos preguntarnos: *¿No hubiese sido más fácil especificar como Date el tipo de dato que contendrá la ProtoVariable "*time*"?* La respuesta es sí, pero esto no es posible. Como hemos dicho en el capítulo anterior, los archivos NetCDF soportan los tipos de datos: ***boolean, byte, char, short, int, long, float, double*** y ***String.*** Como vemos, *Date* no forma parte de ellos.

Por este motivo deberemos convertir los datos tipo *Date* que obtenemos de la base de datos a alguno de los tipos del listado anterior. La primera opción que nos planteamos es convertirlos a tipo *String*, ya que la fecha está formada por números y distintos

caracteres y podríamos tratarla como una cadena de caracteres. Desgraciadamente esta opción no es válida, ya que al convertir los datos a una cadena de caracteres, estos perderían por completo su tipo de dato y el visualizador de datos con el que leeremos los archivos una vez creados, no sería capaz de interpretar dichas cadenas como si fuese una fecha.

Descartada por completo la opción anterior, debemos buscar la manera de representar una fecha con alguno de los restantes tipos de datos compatibles con los archivos. Pues bien, la solución a nuestro problema está en convertir los tipos de datos *Date* a *Double*, ya que Java dispone de funciones para este propósito.

Primeramente, necesitamos establecer qué formato tendrá nuestra fecha, la cual deberá ser exactamente igual al formato de fecha que se encuentra en la base de datos. Para ello utilizaremos *DateFormat*, que nos da esta posibilidad.

```
DateFormat formatDate;
formatDate = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
```

Se ha creado un objeto "formatDate" tipo DateFormat, al cual se le ha especificado el tipo de fecha "yyyy-MM-dd HH:mm:ss" (año-mes-Día hora: minutos: segundos), que será el encargado de indicar el formato de fecha que deberán adoptar los datos que iremos recogiendo de la base de datos.

Obtendremos los datos de la base de datos del siguiente modo:

```
java.util.Date time;
time = formatDate.parse(result.getString("date_sistema"));
```

Los datos que recogemos de *result* los obtenemos tratándolos como una cadena de caracteres. Estos datos los pasamos a *String* porque, por defecto, se encuentran en formato *sql.Date*, ya que se trata de un tipo fecha almacenado en una base de datos. El no hacer este paso nos podría haber dado algún problema porque *sql.Date* y *java.Date* son incompatibles.

Seguidamente, a esta cadena, se le da el formato de fecha (*formatDate*) que hemos especificado y la guardamos en el objeto *time* de tipo java.util.Date. Una vez tenemos nuestra fecha guardada, java.util.Date tiene la función *getTime()* que nos permitirá convertir los objetos tipo *Date* en objetos tipo *Double*.

```
double longTime = time.getTime()/1000;
```

La función *getTime()* nos devuelve la fecha que le pasamos, en este caso "time", en el número de milisegundos que han transcurrido desde el 1 de Enero de 1970. Por eso se almacena en una variable tipo *Double*, ya que se trata de un número muy grande. Como en este caso lo que deseamos es obtener el número de segundos que han transcurrido, dividiremos el valor entre 1000.

Una vez hecho esto, la variable "longTime" almacenará una fecha determinada en formato *double*, ahora sí, compatible con nuestro formato de archivos NetCDF y la podremos añadir a la lista que hemos creado anteriormente para ir almacenando todas las fechas que obtengamos de la consulta SQL realizada a la base de datos.

Para añadir nuevos datos a la lista, solamente tendremos que escribir el siguiente código:

```
dateList.add(longTime);
```

El cual repetiremos mientras *result*, variable donde se guardan los resultados obtenidos de la consulta, contenga información, es decir:

```
while (result.next()){
```

De este modo guardaremos todas las fechas en la lista. Una vez hecho, deberemos cerrar la conexión con la base de datos, utilizando la siguiente función:

```
myCon.close();
```

6.2.5 Inserción de datos en el archivo

Una vez tenemos todos nuestros datos en sus respectivas listas u objetos correspondientes, el siguiente paso será añadir cada listado de datos a su respectiva ProtoVariable, la cual será su contenedor.

Debemos añadir todos los componentes de las listas, es decir, todos los datos a objetos *MultiArray* de las bibliotecas NetCDF. En este caso veamos cómo vamos a introducir los datos que contiene la lista "dateList" en el objeto "timeMa" de tipo *MultiArray*.

```
MultiArray timeMa = new ArrayMultiArray(dateList);
```

Una vez ya hemos insertado los datos en los objetos *MultiArray* vamos a ver cómo vamos a indicar a la aplicación a qué ProtoVariable corresponde. Primeramente, en un nuevo objeto tipo *Variable* llamada "time", vamos a almacenar el nombre, asignado en la declaración, de la ProtoVariable requerida.

```
Variable time = nc.get(timeP.getName());
```

Declararemos un "array" de tipo *int*, llamado "origin", donde indicaremos el número de posiciones iniciales que tendrá nuestro archivo dependiendo del número de dimensiones del mismo. En este caso, al solamente tener una dimensión ("time"), éste constará solamente de una posición.

```
int [] origin = {0};
```

Después, procedemos a la copia de los datos, indicando a qué variable corresponde (en este caso "time"), especificando el origen a partir del cual se deben copiar dichos datos y por último, pasando por parámetro el *MultiArray* ("timeMa"), contenedor de los datos.

```
time.copyin( origin , timeMa);
```

Una vez hayamos realizado este proceso para cada una de las ProtoVariables, deberemos cerrar el archivo (nc) que permanece abierto desde el momento de su creación.

```
nc.close();
```

Cerrando el archivo con esta función, finaliza el proceso de creación de un archivo NetCDF.

6.3 Mejora de la aplicación

En los apartados anteriores hemos visto una parte importante del proceso de desarrollo de una aplicación para generar archivos NetCDF. Con la implementación llevada a cabo, el resultado es la obtención de un archivo estático, que no podemos configurar a nuestro gusto ni según nuestras necesidades. Al ejecutar la aplicación, siempre generará un archivo que contendrá todas las variables del instrumento correspondiente a la tabla en la que hemos realizado la consulta. Como es evidente, aun generando correctamente este tipo de archivos, la aplicación no nos sirve de mucho si no tenemos la posibilidad de escoger qué variables quiero incluir en mi archivo, escoger el instrumento de procedencia de los datos y, por supuesto, durante qué periodo de tiempo quiero que me sean mostradas.

Por estos motivos, en este apartado se mostrará qué modificaciones tendremos que hacer a la versión de la aplicación implementada anteriormente para que cumpla con todos los requerimientos que necesitamos y que hemos especificado en capítulos anteriores.

6.3.1 Selección de Instrumentos

A la hora de personalizar los archivos, lo primero que necesitamos es tener la opción de escoger el instrumento del que recogeremos toda la información. Así pues, utilizaremos una lista desplegable, como la que hemos visto en el apartado de la interfaz de la aplicación, que nos mostrará el listado de instrumentos disponibles.

Para ello, deberemos realizar una consulta SQL en la tabla donde tenemos el listado de instrumentos, "obsea_instrument", y que nos devuelva un listado con el nombre de cada uno de ellos.

```
String sql = "SELECT tipo_instrumento, modelo FROM obsea_instrument";
```

Realizando esta consulta conseguiremos el resultado siguiente:

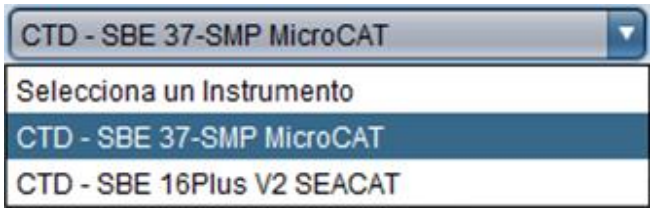


Figura 6.2: Lista de Instrumentos

Una vez seleccionamos el instrumento deseado, queremos que se nos muestre una lista con todas las variables disponibles del instrumento. Para ello, tendremos que realizar la siguiente consulta a la base de datos:

```
String sql = "SELECT date_sistema, temperatura, conductivitat,"
        + "pressio, salinitat, velocitat_so FROM " + Instrument;
```

Donde la variable "Instrument" contiene el objeto seleccionado anteriormente en la lista de instrumentos como vemos a continuación:

```
String Instrument = (String)jSelectInstrument.getSelectedItem();
```

Para aclarar la anterior línea de código, explicaremos que obtenemos un elemento seleccionado llamando a la función *getSelectedItem()* de un objeto en concreto, en este caso del objeto *jSelectInstrument*, que es el nombre asignado a la lista que nos muestra las variables de las que disponemos.

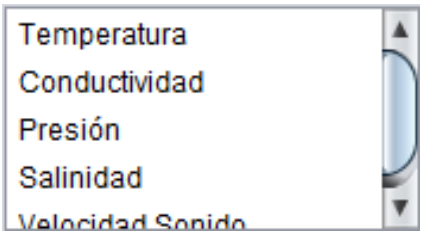


Figura 6.3: Lista de variables

6.3.2 Selección de Variables

También queremos tener la opción de poder escoger el número de variables que desearemos incorporar a los archivos y no, que cada uno de ellos, esté compuesto por todas las variables, ya que según nuestras necesidades y dependiendo del momento, querremos analizar unas u otras. Para ello, dispondremos de otra lista, objeto al que llamaremos *jVarsSelected*, donde iremos añadiendo las variables que queramos.

Este objeto, inicialmente y antes de que empecemos a añadir nuestras variables, no contendrá nada, por lo que le declararemos y añadiremos un modelo de lista, con la función *DefaultListModel()*, que por defecto estará vacío.

```
DefaultListModel modelVarsSelected = new DefaultListModel();
jVarsSelected.setModel(modelVarsSelected);
```

El objeto que se encargará de ir añadiendo cada una de las variables a esta lista será el botón "Añadir". Así que cada vez que hagamos clic en él, deberemos añadir un elemento a la lista *jVarsSelected*, en este caso, el elemento seleccionado de la lista *jVariableList*. Veamos a continuación la manera de lleva a cabo esta acción:

```
String selectedVar = (String)jVariableList.getSelectedValue();
modelVarsSelected.addElement(selectedVar);
```

En estas dos líneas de código vemos cómo primero seleccionamos la variable seleccionada con el método visto anteriormente *getSelectedValue()* y seguidamente lo añadimos a la lista.

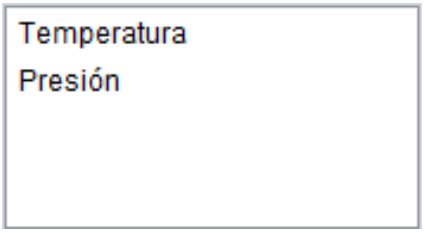


Figura 6.4: Lista de variables añadidas

Para complementar un poco esta parte de la aplicación, por si en algún momento queremos borrar alguna de las variables que previamente hemos introducido por error, también se ha añadido un botón "Eliminar" que nos puede dar solución a este problema. El procedimiento de borrado de la variable se producirá del mismo modo que el procedimiento "Añadir", solo que en lugar de invocar la función *addElement()*, llamaremos al método *remove()*;

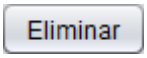


Figura 6.5: Botón "Eliminar"

6.3.3 Intervalo de tiempo

Por último, nos queda escoger el periodo de tiempo para el que queremos visualizar la información escogida en los pasos previos. Para ello, tendremos dos elementos "Fecha Inicial" y "Fecha Final", ambos campos de texto, donde introduciremos las fechas que deseemos.

Un aspecto muy importante a tener en cuenta a la hora de introducir las fechas (inicial y final), serán las dos fechas límite, que automáticamente, nos mostrará la aplicación una vez hayamos escogido un instrumento del listado. Estas, harán referencia sobre a partir de qué fecha y hasta qué fecha disponemos de datos sobre este instrumento. Por este motivo, en ningún caso la "Fecha Inicial" podrá ser inferior a la mostrada por la aplicación y la "Fecha Final" tampoco podrá ser superior a fecha correspondiente también mostrada. En ambos casos no habrá disponibilidad de datos.

Para localizar ambas fechas límite en la base de datos, lo único que se hace es una consulta SQL donde se ha seleccionado primer y último registro de la tabla del instrumento indicado.

Figura 6.6: Parte de la interfaz de la aplicación

En la *figura 6.6*, como ejemplo, hemos llamado a nuestro archivo "temp_pres_ejemplo" y hemos rellenado los campos "Fecha Inicial" y "Fecha Final" con fechas comprendidas a las mostradas por el programa.

Una vez completados todos los campos, procederemos a generar el archio NetCDF, del instrumento seleccionado, las variables requeridas y el periodo de tiempo deseado. Si la creación se produce de forma satisfactoria, se mostrará el siguiente mensaje:

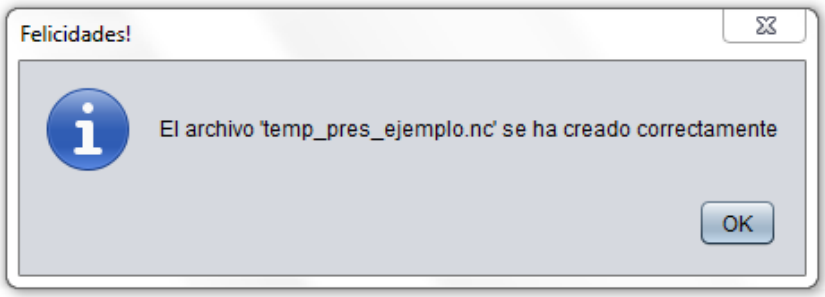


Figura 6.7: Mensaje mostrado por la aplicación

Contrariamente, si no se ha podido generar el NetCDF por algún motivo, nos aparecerá un mensaje similar indicando por qué se ha producido un error y no se ha podido crear el archivo. Alguno de los posibles errores a la hora de generar el archivo podría producirse al no haber añadido ninguna variable o al introducir fechas equivocadas.

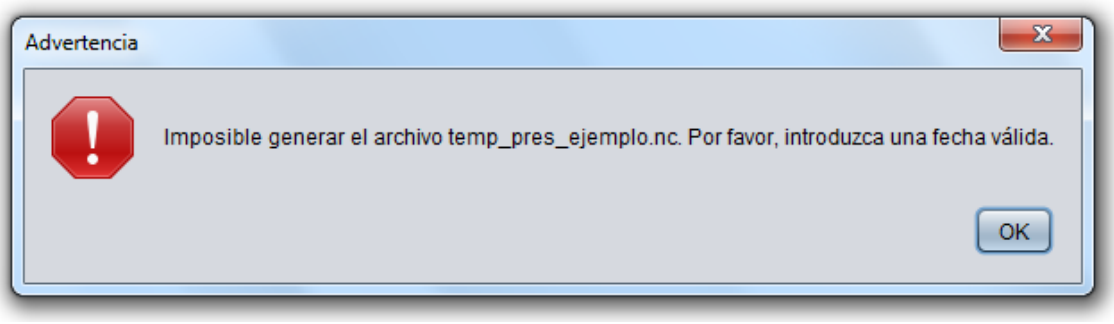


Figura 6.8: Mensaje mostrado por la aplicación

7. Resultados

Una vez ya tenemos el archivo NetCDF creado correctamente, este aparecerá en el directorio especificado por defecto y estará listo para darle utilidad.

Antes de proceder a una visualización gráfica del mismo, echaremos un vistazo a la estructura de su contenido para verificar que realmente contiene los datos que nosotros queremos.

Para ello, se utilizará una herramienta muy simple y muy fácil de instalar en cualquier equipo, ya que se trata de un "Add-In" (complemento) para "Microsoft Excel", una de las herramientas más utilizadas en todo el mundo. Esta aplicación me ha sido de gran utilidad a lo largo del desarrollo de la aplicación para ir comprobando constantemente el contenido de los archivos generados y los resultados obtenidos en cada una de las pruebas realizadas.

Esta herramienta la podremos descargar desde la siguiente dirección URL:

http://code.google.com/p/netcdf4excel, donde accediendo a la pestaña *downloads*, nos mostrará las diferentes versiones del complemento para que lo podamos descargar.

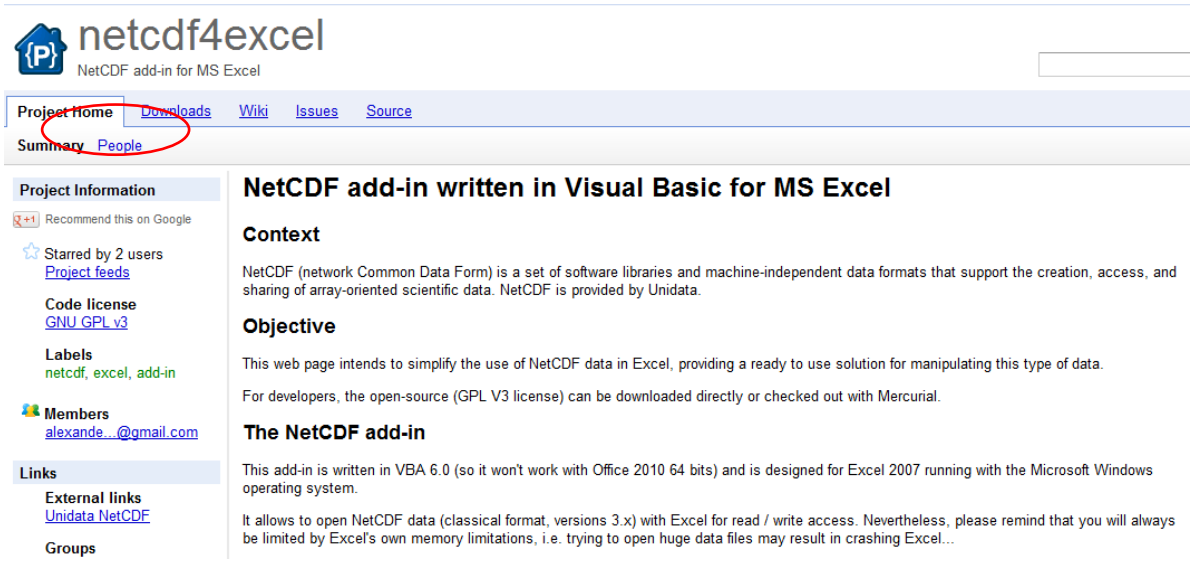


Figura 7.1: Visualización de la página

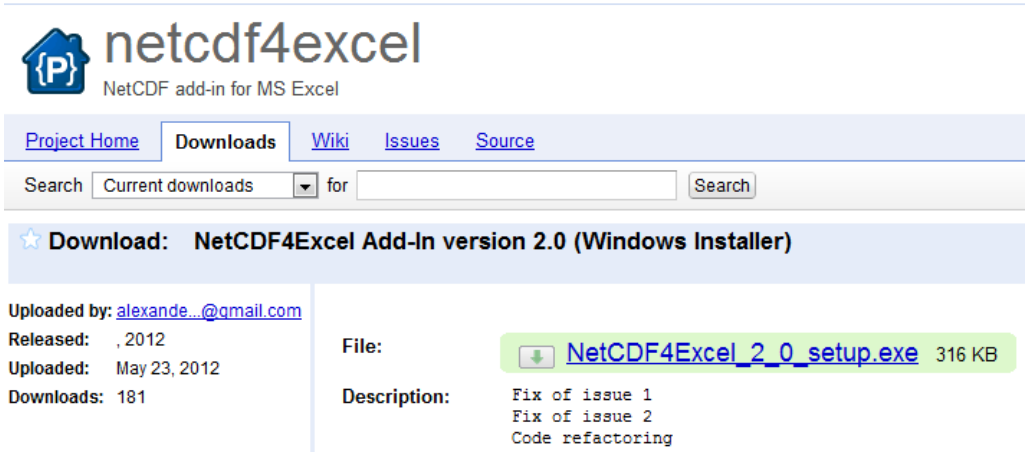


Figura 7.2: Área de descarga del complemento

Una vez descargado, nos aparecerá el complemento en el escritorio, como si de un documento Excel se tratara listo para su ejecución.

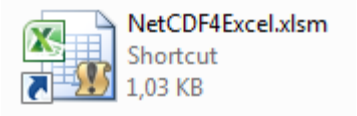


Figura 7.3: Icono del complemento

Al hacer clic sobre él y ejecutarlo, se nos abrirá una ventana de Microsoft Excel totalmente normal, con la única diferencia de que en la parte superior de la interfaz de la aplicación nos aparecerá una nueva pestaña llamada "Add-Ins", donde a través de ella, se puede acceder a cualquiera de los complementos que hayamos instalado.

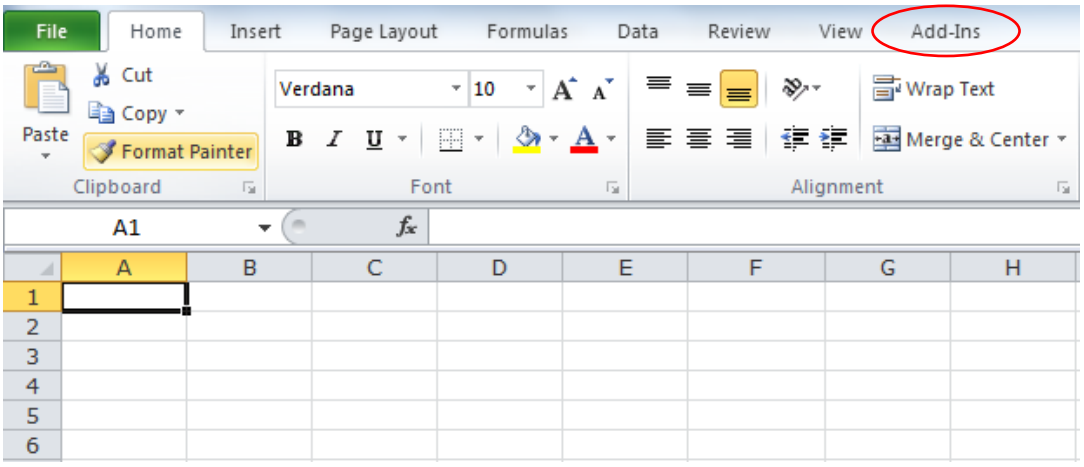


Figura 7.4: Ubicación del complemento

El complemento nos proporciona un pequeño menú desplegable con algunas de las opciones para trabajar con este tipo de archivos. Podemos abrir un archivo y ver la totalidad de su contenido, ver alguna de las variables que contiene o solamente los atributos globales que nos describen la información del mismo.

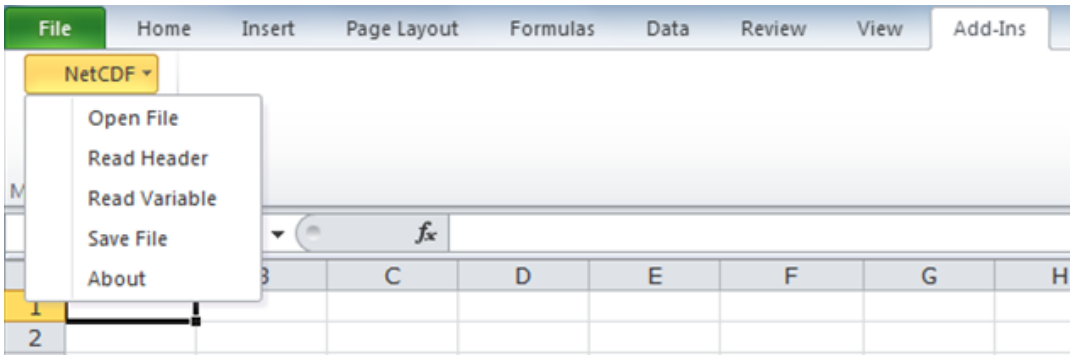


Figura 7.5: Menú del complemento NetCDF para Excel

En este caso, lo que queremos es visualizar la totalidad del contenido del archivo NetCDF, "temp_pres_ejemplo.nc", que hemos creado anteriormente. Una vez abierto, se nos mostrarán dos hojas de cálculo, una con la información del contenido del archivo y otra con todos los datos que este contiene.

Inicialmente, en la primera hoja ("NC_Info"), se nos mostrarán los atributos globales, las dimensiones y las variables que contiene el archivo.

En primer lugar aparecerán los atributos globales del archivo encargados de darnos una descripción general del contenido del archivo:

	A	B	C	D	E	F	G	H
4	Global Attributes							
5	Name	NcType	Value					
6	title	NC_CHAR	"Mooring OBSEA CTD data from SARTI"					
7	creator_name	NC_CHAR	"Departament of Computer Technology and Communication"					
8	Institution	NC_CHAR	"Universitat Politécnica de Catalunya"					
9	Platform	NC_CHAR	"OBSEA Expandable SeaFloor"					
10	CTD_Instrument	NC_CHAR	"SBE 16Plus V2 SEACAT"					
11	date_created	NC_CHAR	"Sun April 01 2012"					
12	time_coverage_start	NC_CHAR	"2012-04-01 00:00:00"					
13	time_coverage_end	NC_CHAR	"2012-04-05 23:59:59"					
14	netcdf_version	NC_CHAR	"4.2"					
15	Convention	NC_CHAR	"CF-1.6"					
16	keywords	NC_CHAR	"Ocean;Moorings;Observatory"					

Figura 7.6: Listado de Atributos Globales

De ellos, podemos obtener información valiosa que nos ayudarán a interpretar los datos y ser conocedores de su procedencia.

Seguidamente, se nos mostrarán las dimensiones del archivo, en este caso y como bien hemos dicho antes, solamente contendrá una de ellas, la dimensión tiempo ("time").

18	Dimensions		
19	Name	Excel Nam	Value
20	time	time	43182
21			

Figura 7.7: Listado de Dimensiones

En este caso, el valor que se le asigna a "time" hace referencia al número de líneas de datos que el archivo contiene, es decir, al número total de datos que contiene cada una de las variables. Como podemos ver, cada una de las variables contiene 43.182 registros.

Por último, a continuación de las dimensiones, nos mostrará un listado de todas las variables que contiene el archivo.

22	Variables							
23	Name	Excel Nam	NcType	Dimensio	Attribute Name	Attribute Nc	Attribute Value	NcStorageType
24	time	time	NC_DOUB	time	long_name	NC_CHAR	"time"	NC_DOUBLE
25	-	-	-	-	standard_name	NC_CHAR	"time"	-
26	-	-	-	-	axis	NC_CHAR	"T"	-
27	-	-	-	-	calendar	NC_CHAR	"gregorian"	-
28	-	-	-	-	time_zone	NC_CHAR	"CET"	-
29	-	-	-	-	units	NC_CHAR	"seconds since 1970-	-
30	latitude	latitude	NC_DOUB	time	standard_name	NC_CHAR	"latitude"	NC_DOUBLE
31	-	-	-	-	units	NC_CHAR	"degrees_north"	-
32	-	-	-	-	axis	NC_CHAR	"Z"	-
33	longitude	longitude	NC_DOUB	time	standard_name	NC_CHAR	"longitude"	NC_DOUBLE
34	-	-	-	-	units	NC_CHAR	"degrees_east"	-
35	-	-	-	-	axis	NC_CHAR	"Z"	-
36	depth	depth	NC_DOUB	time	standard_name	NC_CHAR	"depth"	NC_DOUBLE
37	-	-	-	-	units	NC_CHAR	"m"	-
38	-	-	-	-	axis	NC_CHAR	"Z"	-
39	temperature	temperati	NC_DOUB	time	long_name	NC_CHAR	"water temperature	NC_DOUBLE
40	-	-	-	-	standard_name	NC_CHAR	"sea_water_temper	-
41	-	-	-	-	units	NC_CHAR	"Celsius"	-
42	-	-	-	-	axis	NC_CHAR	"Z"	-
43	pressure	pressure	NC_DOUB	time	long_name	NC_CHAR	"water pressure"	NC_DOUBLE
44	-	-	-	-	standard_name	NC_CHAR	"sea_water_pressur	-
45	-	-	-	-	units	NC_CHAR	"decibar"	-
46	-	-	-	-	axis	NC_CHAR	"Z"	-

Figura 7.8: Listado de Variables

De cada una de las variables que vemos en la lista anterior, se nos indica el número de atributos que contiene, el valor de cada uno de los atributos y el tipo de dato que contiene. Una descripción rápida y eficaz del contenido general del archivo.

Para ver los datos que contiene cada variable, cambiaremos a la segunda hoja que se nos ha creado, "time". El nombre corresponde a la dimensión a la que pertenecen dichos datos. Por ello, se creará una hoja para cada una de las dimensiones de las que dispongamos, en este caso al solamente tener una dimensión, se creará una hoja adicional (además de "NC_Info") que siempre formará parte de la información del archivo.

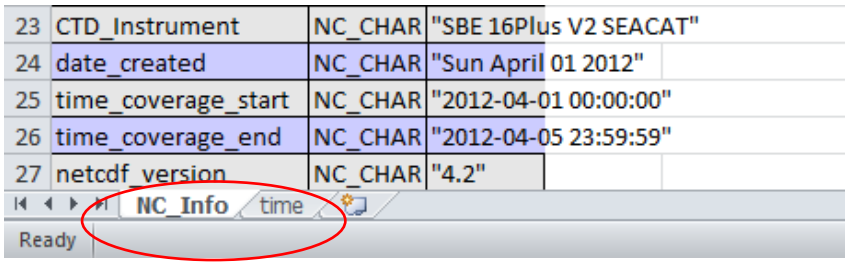


Figura 7.8: Hojas del documento

Veamos qué datos contiene la dimensión "time":

	time	latitude	longitude	depth	temperature	pressure
2	1333231200	41,1819083	1,7523417	19,32	13,5189	19,36
3	1333231210	41,1819083	1,7523417	19,32	13,519	19,359
4	1333231221	41,1819083	1,7523417	19,32	13,52	19,359
5	1333231230	41,1819083	1,7523417	19,32	13,519	19,358
6	1333231240	41,1819083	1,7523417	19,32	13,5174	19,369
7	1333231250	41,1819083	1,7523417	19,32	13,5169	19,355
8	1333231260	41,1819083	1,7523417	19,32	13,5172	19,359
9	1333231270	41,1819083	1,7523417	19,32	13,517	19,354
10	1333231280	41,1819083	1,7523417	19,32	13,5171	19,354
11	1333231290	41,1819083	1,7523417	19,32	13,5171	19,366
12	1333231300	41,1819083	1,7523417	19,32	13,5174	19,361
13	1333231310	41,1819083	1,7523417	19,32	13,5175	19,36
14	1333231320	41,1819083	1,7523417	19,32	13,5172	19,362
15	1333231330	41,1819083	1,7523417	19,32	13,5174	19,367
16	1333231340	41,1819083	1,7523417	19,32	13,5179	19,364
17	1333231350	41,1819083	1,7523417	19,32	13,5166	19,363
18	1333231360	41,1819083	1,7523417	19,32	13,5171	19,363
19	1333231370	41,1819083	1,7523417	19,32	13,518	19,36
20	1333231380	41,1819083	1,7523417	19,32	13,5169	19,363
21	1333231390	41,1819083	1,7523417	19,32	13,5169	19,368
22	1333231400	41,1819083	1,7523417	19,32	13,5174	19,363
23	1333231410	41,1819083	1,7523417	19,32	13,519	19,359
24	1333231420	41,1819083	1,7523417	19,32	13,5227	19,365
25	1333231430	41,1819083	1,7523417	19,32	13,5232	19,363
26	1333231440	41,1819083	1,7523417	19,32	13,5244	19,37

Figura 7.8: Datos que contiene el archivo

Se nos muestran todos los datos que contiene el archivo fragmentados por las variables a las que corresponde cada tipo de información. Como se ve en este caso, el NetCDF dispone de seis variables: ***time, longitude, latitude, depth, temperature, pressure***.

Longitude, latitude y *depth* nos describen la ubicación de dónde proceden los datos. Al tratarse de un observatorio submarino, en ningún momento variará su ubicación, por eso todos los campos de las respectivas variables contienen el mismo dato, el cual hemos recogido de la tabla "obsea_instrument" donde se almacena la información sobre los instrumentos.

Podríamos pensar que es una tontería llenar todos los campos con una misma variable, ya que en este caso se trata de un dato estático porque nunca variará. Pero es importante saber la ubicación de cada dato obtenido porque, en un futuro, se podrían generar archivos NetCDF's para algún tipo de vehículo que estuviera constantemente en movimiento. En ese caso, tanto *longitude, latitude* y *depth* podrían cambiar de valor en cada instante.

Temperature y *pressure* nos muestra los datos que hemos ido introduciendo procedentes de su correspondiente tabla en la base de datos. La variable *time*, contenedor de la fecha en la que obtuvieron los datos, nos muestra sus datos en formato numérico, por la conversión realizada durante la aplicación, ya que estos datos debían estar en alguno de los formatos soportados por los archivos NetCDF. Si nos fijamos, estos números grandes que representan la fecha, van incrementando su valor en 10, debido a que el instrumento CTD nos proporciona datos cada 10 segundos.

8. Visualización de los datos

Hemos generado un archivo NetCDF y mediante la utilidad, vista en el capítulo anterior, hemos podido comprobar que la información y su contenido correspondiente son los deseados.

En este capítulo vamos a ver el contenido del archivo NetCDF de una forma más gráfica gracias a *Ocean Data View* (ODV).

Este visualizador es un paquete de software para la exploración, análisis y visualización de datos oceanográficos, de manera muy ágil, con series de tiempo, trayectoria o secuencia. Es una aplicación totalmente compatible con los Sistemas Operativos Windows, Mac OS, Linux y UNIX.

Gracias a este software, una gran variedad de datos pueden ser fácilmente explorados en el escritorio de cualquier computadora. Por supuesto, ODV permite explorar y visualizar los archivos NetCDF y el vocabulario estándar CF.

El software puede ser libremente descargado desde su sitio web oficial. Eso sí, previamente deberemos habernos registrado para tener la posibilidad de hacer la descarga. Una vez descargada la aplicación, podremos ejecutarla y proceder a la visualización del archivo que deseemos en cada momento.

Primeramente, deberemos escoger el archivo que queremos visualizar. El primer paso que se nos pedirá, será la elección de la dimensión para la que queremos ver el archivo porque dependiendo de la que escojamos, la visualización de los datos será una u otra. En este caso, como vemos en el listado, nuestro archivo solamente contiene una dimensión.

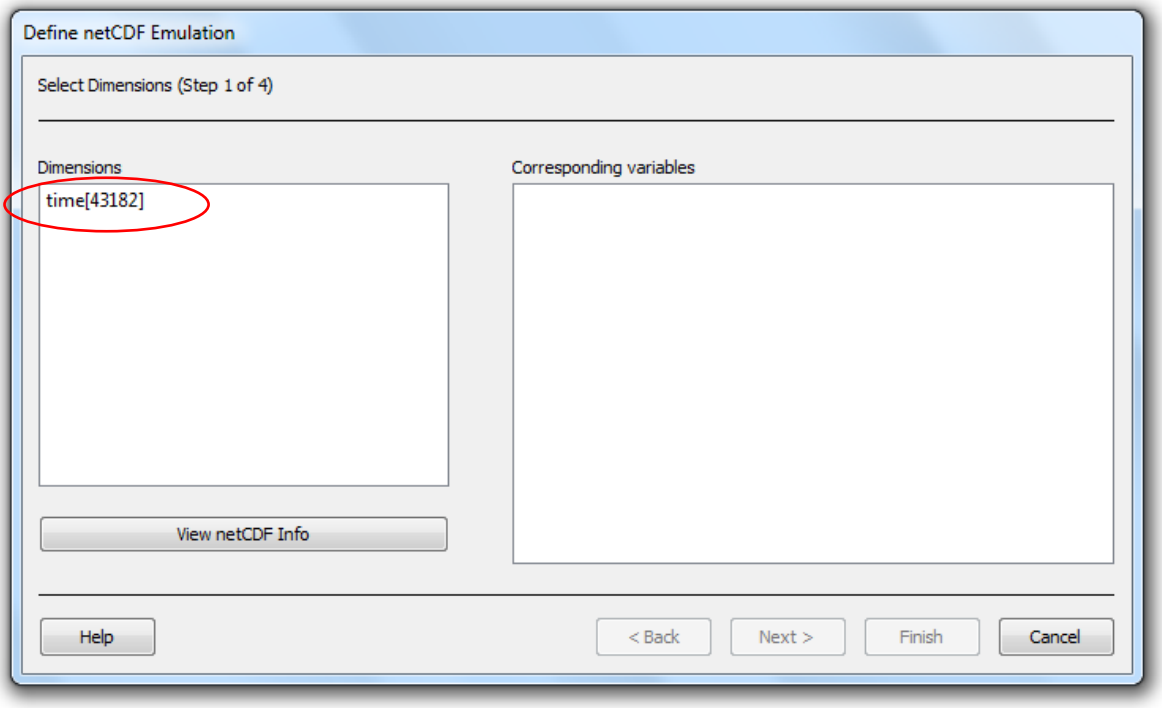


Figura 8.1: Paso 1

Antes de seleccionarla y pasar al siguiente paso, en la parte inferior vemos un botón llamado "View netCDF Info" donde se nos abrirá un archivo de texto y nos mostrará una lista con un sumario del contenido del archivo. Variables, Dimensiones y Atributos.

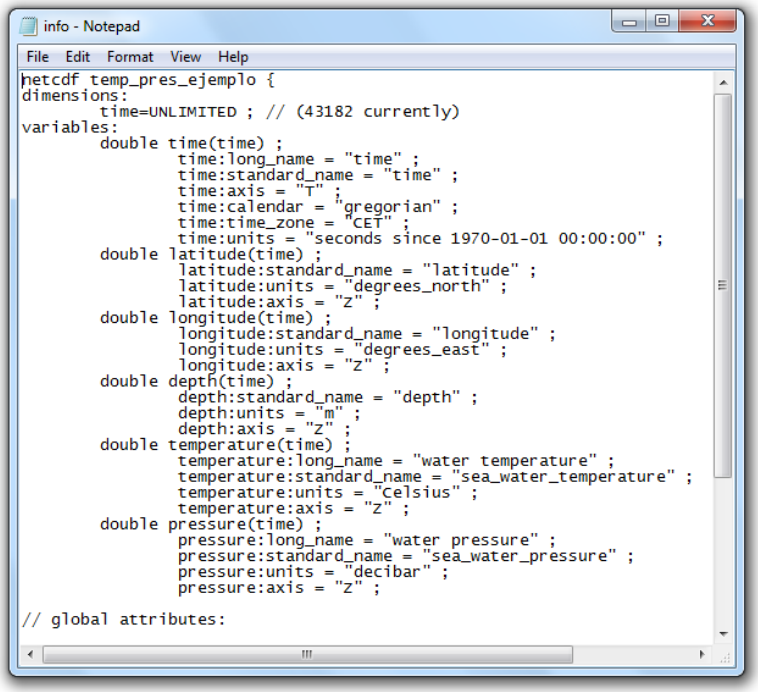


Figura 8.2: Archivo de información

Si pasamos a la siguiente ventana, aparecerán dos listas. La lista de la izquierda “Source Variables” nos muestra todas las variables que contiene el archivo. En la de la derecha, “Meta Variables”, se muestra una lista de algunas de las meta-variables más comunes en este tipo de archivos que ayudan a describirlo y a explicar su contenido. Se marcarán con un asterisco todas las meta-variables disponibles en el NetCDF.

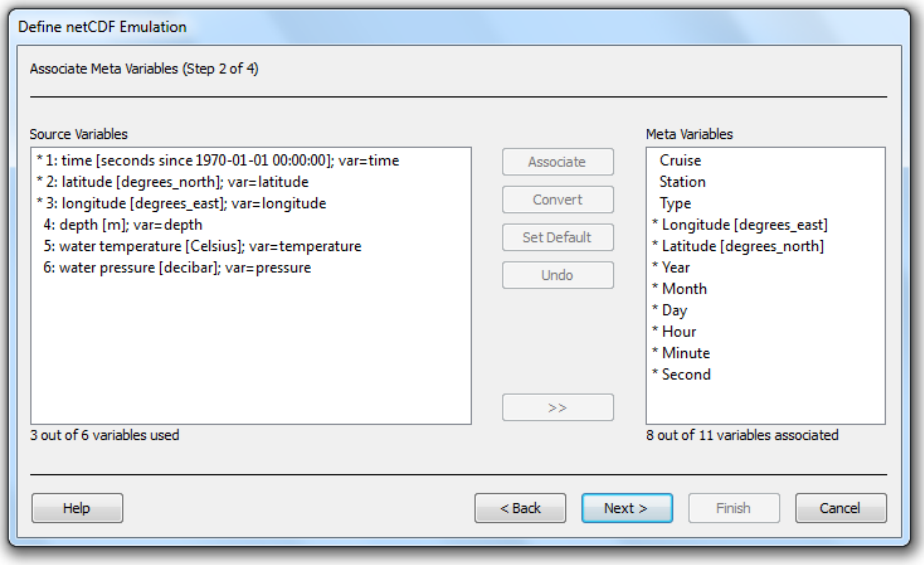


Figura 8.3: Paso 3

El último paso será escoger, en el caso de haber varias estaciones produciendo datos, la deseada y darle a “Finalizar”. Todas ellas se mostrarán en el mapa que vemos en la parte de la derecha gracias a la introducción de las meta-variables *longitude* y *latitude* que las situarán en sus coordenadas exactas.

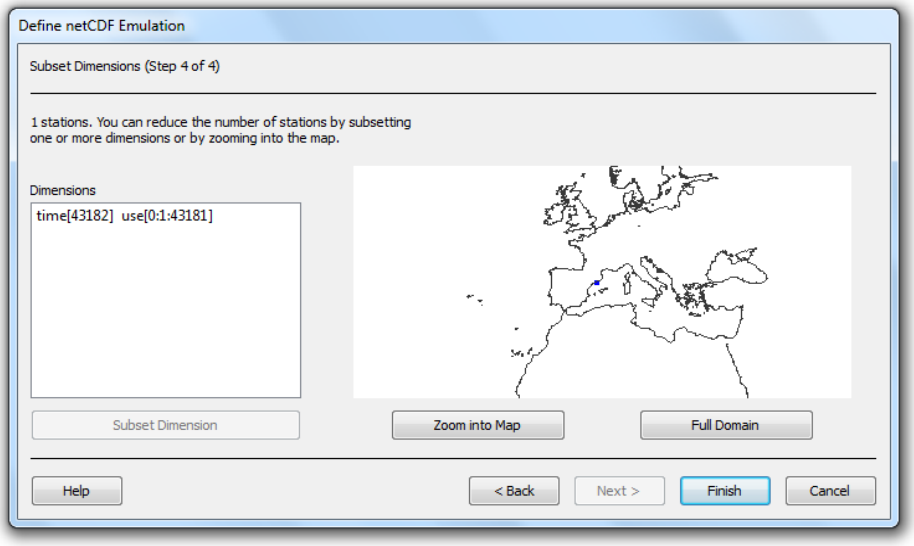


Figura 8.4: Paso 4

Esta será la vista general de toda la información que nos mostrará el visualizador:

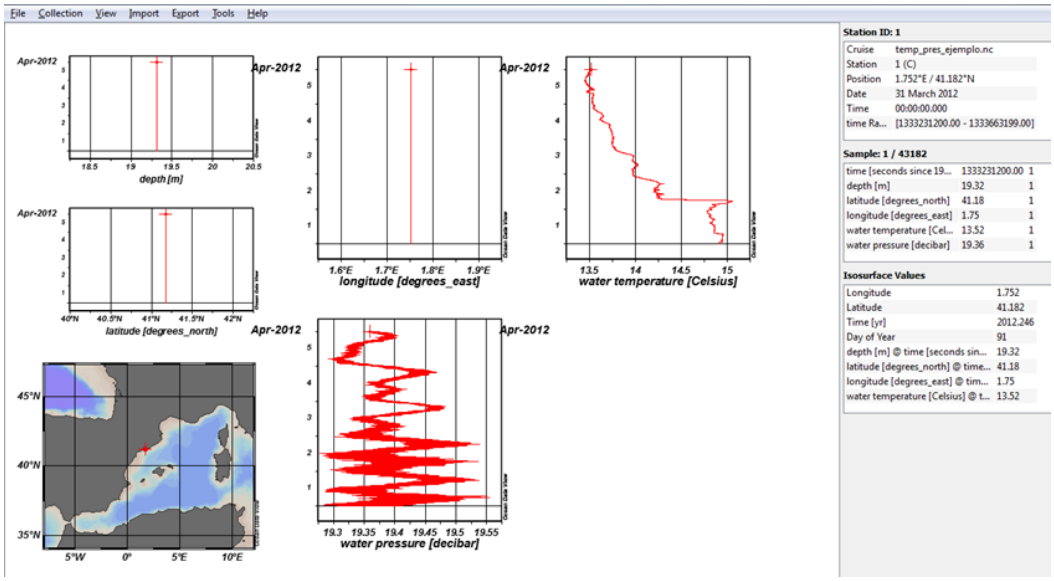


Figura 8.5: Visualización gráfica general

El panel principal contiene todas las gráficas generadas para cada una de las variables y un mapa donde podemos ver marcada la ubicación del observatorio. En la parte derecha del visualizador, se muestra un menú lateral con los datos numéricos de cada una de las variables de las diferentes muestras obtenidas.

Fijémonos que los tres gráficos de las variables *longitude*, *latitude* y *depth* están representados por una línea recta continua. Eso es debido a que a lo largo del documento su valor permanece inalterado.

Al ampliar la gráfica donde se ubica la procedencia de los datos, vemos con más claridad y más precisión las coordenadas de su situación (según los ejes X, Y) como se comprueba en las dos siguientes imágenes.

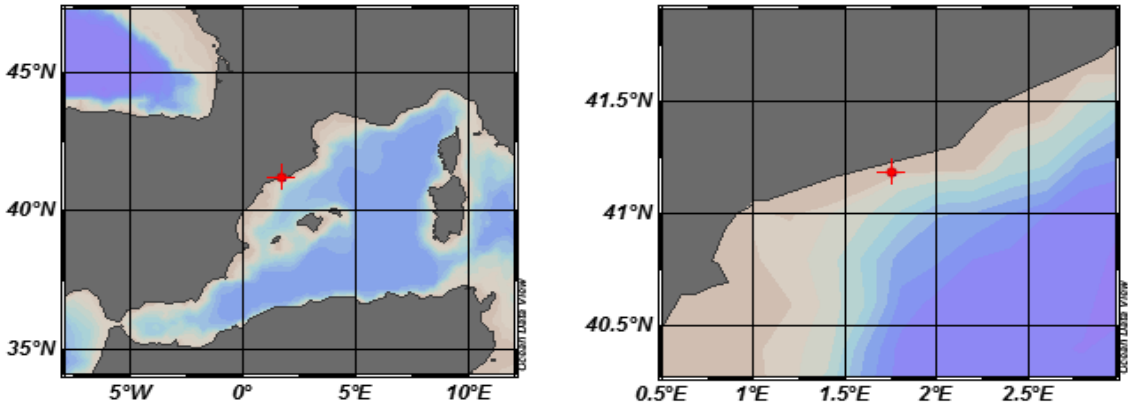


Figura 8.6: Gráfico de la ubicación de los datos

En los gráficos referentes a las variables *pressure* y *temperature*, se nos muestra la progresión de los datos a lo largo del periodo de tiempo establecido previamente a la creación del archivo. En el eje "Y" se puede ver, en ambos casos, como los valores de la escala van del 1 al 5, haciendo referencia a los días del mes correspondiente, en este caso el mes de Abril. Por otra parte, en el eje "X", los valores para determinar la escala son dinámicos y van variando según los datos que contiene cada variable.

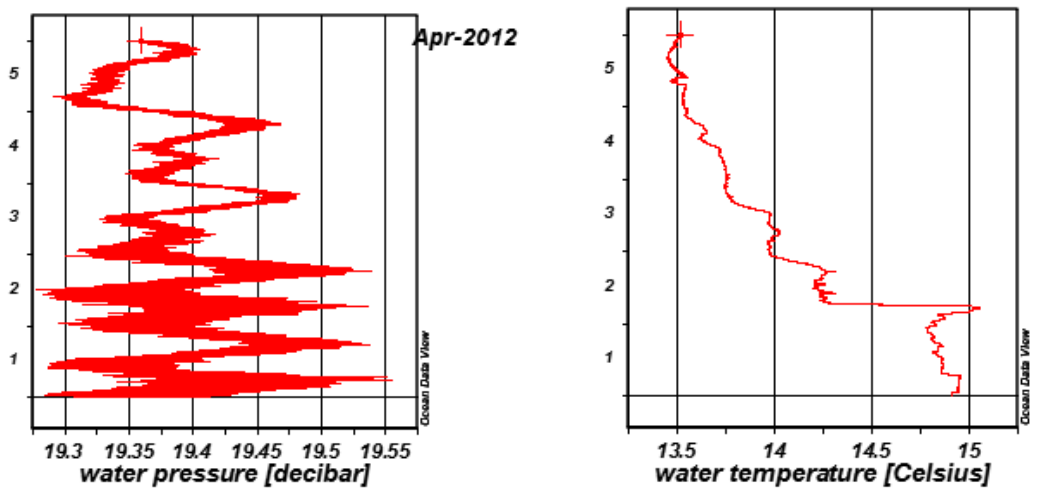
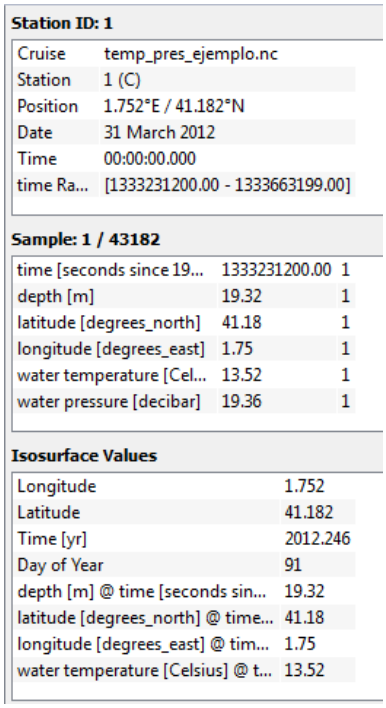


Figura 8.7: Gráficos de presión y temperatura

El último elemento que resta por ver con un poco más de detalle es la barra lateral de la derech



Station ID: 1

Cruise	temp_pres_ejemplo.nc
Station	1 (C)
Position	1.752°E / 41.182°N
Date	31 March 2012
Time	00:00:00.000
time Ra...	[1333231200.00 - 1333663199.00]

Sample: 1 / 43182

time [seconds since 19...	1333231200.00	1
depth [m]	19.32	1
latitude [degrees_north]	41.18	1
longitude [degrees_east]	1.75	1
water temperature [Cel...	13.52	1
water pressure [decibar]	19.36	1

Isosurface Values

Longitude	1.752
Latitude	41.182
Time [yr]	2012.246
Day of Year	91
depth [m] @ time [seconds sin...	19.32
latitude [degrees_north] @ time...	41.18
longitude [degrees_east] @ tim...	1.75
water temperature [Celsius] @ t...	13.52

Figura 8.8: Menú lateral del panel

9. Difusión de los datos

Una vez se dispone de los archivos NetCDF, llega el momento de compartir los datos que se han generado con el resto de la comunidad científica. Para ello, se ha decidido integrar los archivos, que se hayan ido creando con la aplicación, en la red de sistemas de observación **SeaDataNet**.

9.1 SeaDataNet

SeaDataNet es un proyecto mediante el cual se pretende crear una infraestructura Pan-Europea que permita un acceso integrado de los datos, metadatos, productos y servicios que generan los diferentes grupos de investigación oceanográfica de los países asociados.

Existen 35 países que bordean los mares europeos y más de 600 laboratorios científicos que producen datos usando diferentes sensores y plataformas como barcos, boyas, satélites, etc.

Figura 9.1: Países asociados a SeaDataNet

Se pretende establecer la red de Centros de Datos de estos 35 países que sean activos y profesionales en recopilación y análisis de datos con el fin de difundir on-line y con una calidad estandarizada, tanto los datos como los productos generados a partir de ellos.

En la mayoría de los casos, la información generada es incompatible y cada país o cada laboratorio se ve obligado a producir sus propios datos. Este proyecto tiene como objetivo desarrollar una infraestructura común para la gestión de los datos oceanográficos, para lo que se requiere la adopción de unos estándares comunes en la creación y comunicación de toda esta información. Los datos son gestionados por Centros Nacionales de Datos Oceanográficos (NODC) y por los coordinadores de los diferentes centros asociados. La infraestructura creada por SeaDataNet ofrece una plataforma virtual de acceso a toda la información generada a través de su portal único interconectado *www.seadatanet.org*.

SeaDataNet es una infraestructura formada por los siguientes directorios:

European Directory of Marine Organisations (EDMO): EDMO contiene las direcciones y los perfiles de la actividad de los institutos de investigación, centros de datos, organizaciones privadas y gubernamentales que de una manera u otra participan en actividades de investigación oceanográfica y marina. En la actualidad, EDMO contiene más de 1.500 organizaciones.

Su interfaz de consulta permite hacer busquedas por criterios. Las organizaciones que contiene EDMO son mostradas en un mapa, donde se puede acceder a cada una de ellas para obtener información sobre su perfil y dirección. Una vez estamos dentro del perfil de una organización se nos mostrará, mediante iconos, la participación de ésta en cualquiera de los servicios ofrecidos por SeaDataNet o si dispone de servicio de acceso a datos (CDI).

European Directory of Marine Environmental Data sets (EDMED): EDMED es una referencia general a los conjuntos de datos marinos y colecciones de estos realizadas en los laboratorios de investigación europeos, con el fin de proporcionar a los científicos, ingenieros, etc. un mecanismo para hacer simple su identificación.

Abarca una amplia gama de disciplinas, incluyendo la meteorología marina, física, oceanografía química y biológica, sedimentología, biología marina, estudios costeros, etc. Todos ellos se describen en EDMED independiente de su formato (bases de datos digitales o archivos, registro analógicos, gráficos, fotografías y videos, muestras geológicas, etc…) En la actualidad, EDMED describe más de 3.500 conjuntos de datos procedentes de más de 700 centros de toda Europa.

European Directory of Marine Environmental Research Projects (EDMERP): EDMERP cubre proyectos de investigación marina para una amplia gama de disciplinas como las mencionadas anteriormente. Se describen como hojas de datos y metadatos con sus aspectos más relevantes. El objetivo principal es ayudar a los usuarios a identificar las actividades de investigación y a las organizaciones que participan en toda Europa. Actualmente, EDMERP describe más de 1.800 proyectos de investigación de distintas organizaciones europeas.

European Directory of the initial Ocean-observing Systems (EDIOS): EDIOS es el Directorio Europeo de Sistemas de Observación del océano, una única base de metadatos de búsqueda. Este directorio proporciona una nueva herramienta basada en Internet para buscar información sobre los sistemas de observación que operan, con regularidad y de forma rutinaria, en las aguas europeas. En él se contienen los metadatos de los sistemas europeos de observación como plataformas, boyas, etc. EDIOS es una iniciativa del EuroGOOS6 y actualmente contiene más de 12.000 entradas de datos que se actualizan periódicamente.

[6] EuroGOOS: European Global Coean Observing System

9.2 Registro del observatorio OBSEA

El primer paso para poder difundir e integrar los datos, que va generando el observatorio y que posteriormente son manipulados por la aplicación y convertidos a formato NetCDF, es registrar el observatorio OBSEA en la red de sistemas de observación SeaDataNet. Para ello se deberá registrar en el Directorio Europeo de Sistemas de Observación (EDIOS) donde figuran todas las infraestructuras que general algún tipo de información oceanográfica.

Para llevar a cabo el registro del observatorio y de la organización de la cual procede la información generada, se deberá contactar, en este caso, con el Instituto Español de Oceanografía (IEO), responsables del nodo de España en SeaDataNet. Ellos serán los encargados de introducir el observatorio OBSEA y al grupo SARTI-UPC en la red de sistemas de observación a través de los datos que se les envía rellenando un formulario proporcionado por EDIOS.

En este formulario, se introducirá todo tipo de información relacionada con el centro y la plataforma a registrar, el observatorio OBSEA. Deberemos rellenar campos sobre el nombre del centro y de la plataforma.

En relación al centro bastará con especificar el nombre de la organización o institución responsable de la plataforma, país de procedencia, dirección del centro y un responsable de contacto para tratar todo lo relacionado con la plataforma registrada. Respecto al observatorio, se deberá hacer una descripción de las características más destacadas de éste, como por ejemplo el año de fabricación y cuando se puso en funcionamiento, su ubicación, una descripción de la plataforma y, lo más importante, explicar con detalle los instrumentos que lo componen y que se encargar de recolectar la información.

Con esta información se podrá proceder al registro del grupo SARTI-UPC en el directorio EDMO, donde figuran un gran número de organizaciones europeas que comparten la información generada por sus plataformas o instrumentos con el resto de la comunidad científica.

Para el registro del observatorio OBSEA en el directorio EDIOS, será necesario hacer un paso más. Con la información que se ha proporcionado en el formulario sobre la plataforma y los instrumentos, se deberá crear un fichero XML a través de ellos. Afortunadamente, se dispone de un software llamado “MIKADO”, proporcionado por SeaDataNet, que realiza de forma automática el proceso de conversión a este tipo de archivos. Una vez transformada la información a XML se podrá proceder a la carga de este archivo en el directorio EDIOS.

Una vez haya finalizado el registro, el grupo SARTI-UPC aparecerá entre el listado de organizaciones disponibles que están vinculadas en el directorio EDMO.

Para buscar el observatorio o infraestructura deseados, tomaremos como ejemplo el filtro de búsqueda "UPC" y seleccionaremos el país deseado. En la parte inferior nos aparecerá un listado con todos los resultados de nuestra búsqueda.

Nota: El observatorio OBSEA actualmente está en proceso de ser registrado, por este motivo todavía no aparece como uno de los posibles resultados de búsqueda en la lista inferior de la figura 9.2.

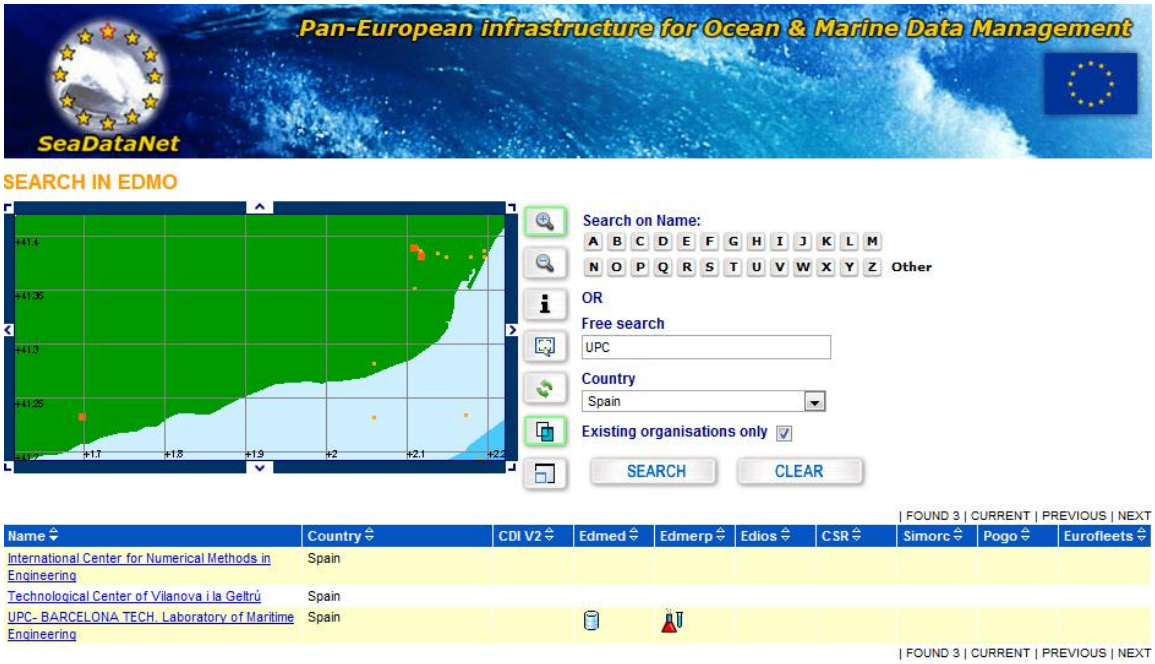


Figura 9.2: Interfaz de búsqueda del directorio EDMO.

9.3 Visualización de datos y metadatos

Para ver con más detalle todas las posibilidades que se nos ofrecen una vez tenemos nuestra plataforma registrada, usaremos como ejemplo de búsqueda el "IEO/Santander Oceanographic Centre", ya que es el encargado de registrar cualquier plataforma de España en la red SeaDataNet y disponen de infraestructuras y datos suficientes que nos servirán para ver, con más claridad, el funcionamiento de esta red de sistemas de observación oceanográfica.

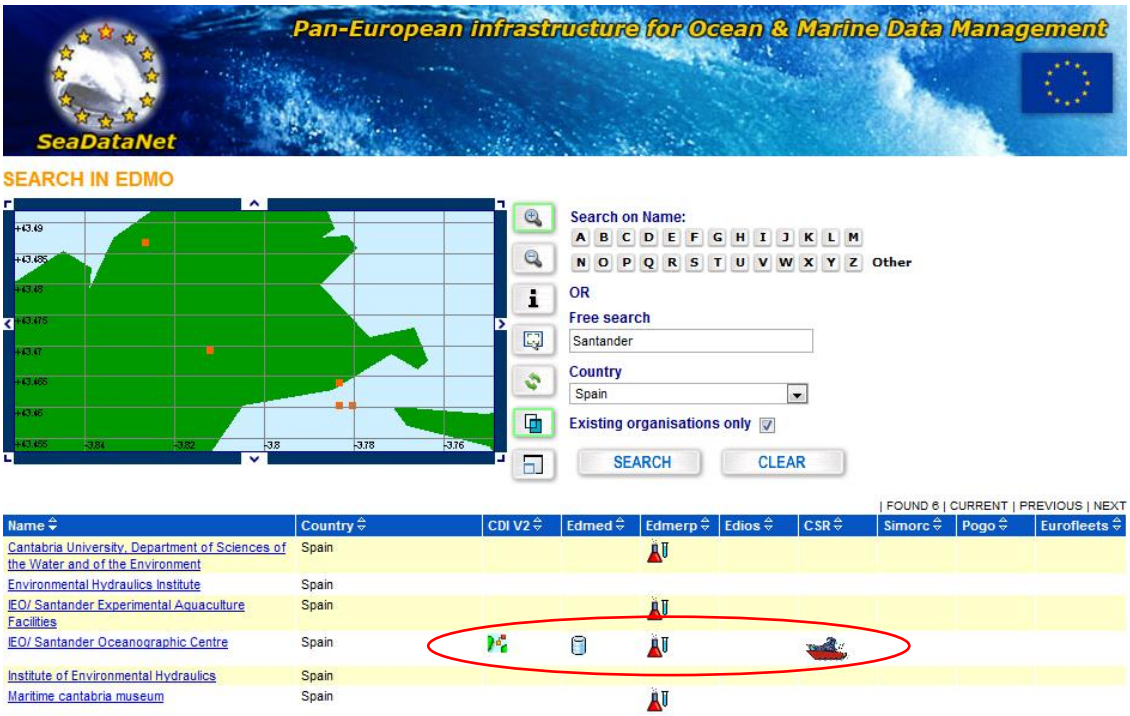


Figura 9.3: Interfaz de búsqueda del directorio EDMO.

Como observamos en la figura 9.3, mediante una serie de iconos, se nos indica el tipo de datos y metadatos disponibles en cada plataforma. En este caso, vemos que en la escogida para nuestro ejemplo (IEO/Santander Oceanogrphic Centre) dispone de cuatro tipos diferentes de datos en sus correspondientes directorios: CDI, Edmed, Edmerp y CSR7.

[7] CSR: Los "Cruise Summary Reports" son informes sobre pruebas o experimentos realizados en el mar.

Para acceder a cada uno de ellos, solamente deberemos hacer clic sobre el icono pertinente. Empezaremos por los datos CDI (Common Data Index).

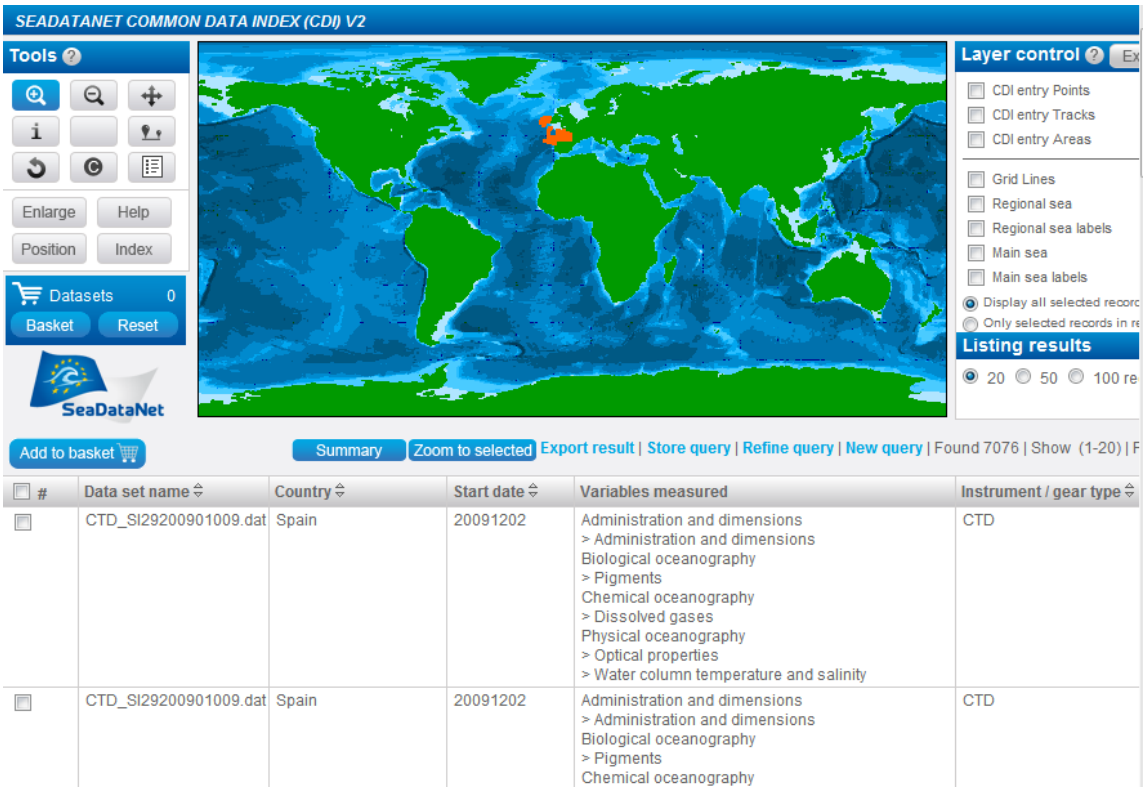


Figura 9.4: Interfaz de visualización de los datos

A través de esta interfaz podremos acceder a todos los archivos de datos disponibles que el centro u organización hayan decidido añadir para ser compartidos con los usuarios de la comunidad de SeaDataNet. Vemos como en la parte inferior se nos muestran los archivos disponibles, indicándonos, gracias a los metadatos que nos detallan la información contenida dentro del archivo, cuál es su nombre, el país de procedencia, la fecha, el tipo de variables e información que nos mostrarán y el instrumento que ha recolectado dichos datos.

Será en esta sección, donde una vez registrado el observatorio OBSEA, se añadirán y serán mostrados todos y cada uno de los archivos NetCDF (archivos que cumplen con el CDI) que se hayan ido generando con la aplicación que se ha desarrollado a lo largo del proyecto.

10. Conclusiones

El objetivo principal de este proyecto ha sido el desarrollo e implementación de una aplicación software capaz de generar archivos con datos y metadatos estandarizados que debía cumplir una serie de requerimientos especificados según un análisis previo realizado anteriormente. Esta aplicación, debía encargarse de generar archivos NetCDF con la información que se recibía procedente de los diferentes instrumentos que componen el observatorio OBSEA, en este caso los CTD's.

Se puede decir que el objetivo ha sido alcanzado, gracias al desarrollo de una aplicación que genera dichos archivos dinámicos con datos e información diferentes dependiendo de la procedencia de estos y las necesidades del usuario.

El otro de los objetivos era incorporar la información generada en la red de sistemas de observación SeaDataNet. Cabe decir que este objetivo no se ha podido completar en su totalidad todavía, ya que el registro del observatorio OBSEA se ha demorado más de lo previsto al no depender de nosotros exclusivamente. Este proceso debe ser realizado por parte del nodo responsable de cada país y a día de hoy todavía no tenemos respuesta de que se haya finalizado el registro del observatorio. Eso sí, podemos decir, con toda seguridad, que es cuestión de tiempo y se producirá en los próximos días.

10.1 Futuras aplicaciones

En la actualidad, la aplicación desarrollada cubre perfectamente las necesidades del centro, ya que es válida para cualquiera de los instrumentos disponibles en el observatorio OBSEA. En un futuro, no es descartable que se desarrollen más nodos como el existente, es decir, el proyecto OBSEA que está desarrollando el grupo SARTI, una vez finalizado estará compuesto por varios nodos (observatorios) y por ello la aplicación deberá adaptarse a esta posible situación.

En esta nueva línea de trabajo que se nos abre, la aplicación deberá adaptarse a las nuevas necesidades del sistema, en este caso deberá poder generar archivos NetCDF para cada uno de los nodos que componen la red de observatorios del proyecto OBSEA.

La adaptación a esta nueva situación no nos será difícil, ya que el grueso del trabajo ya lo tendremos hecho. El procedimiento para crear los archivos será el mismo, solamente deberemos añadir un nuevo elemento, un listado desplegable por ejemplo, donde se nos dé la posibilidad de escoger un observatorio previamente.

Esta será la única modificación de deberemos realizar para que la aplicación se adapte bien a los nuevos requerimientos. Eso sí, previamente deberemos haber registrado los

observatorios y haberlos introducido en nuestra base de datos, así como cada uno de los instrumentos que los componen. Además, en caso que sea necesario, deberemos complementar la tabla donde se almacena la información referente al vocabulario estándar que se debe utilizar para cada uno de los instrumentos y la información que ellos generan.

10.2 Aprendizaje y experiencia personal

En cuanto a mi experiencia personal, pienso que el desarrollo y realización de este proyecto me han aportado muchas cosas positivas que me serán de gran utilidad de cara al futuro y en otros posibles proyectos.

Principalmente, me ha aportado grandes y nuevos conocimientos sobre el tratamiento de datos y ser conocedor de la complejidad de este mundo. Durante este proyecto me he adentrado en un entorno del que conocía muy poco hasta entonces y por ese motivo me resultó difícil asimilar muchos de los conceptos, sobre todo al inicio de este, ya que tuve que documentarme de muchas fuentes para , poco a poco, ir asumiendo dichos conceptos y aclarando ideas.

He aprendido a utilizar la API de programación de NetCDF para Java y mejorar, notablemente, mi nivel de programación para este lenguaje. También cabe mencionar, que me encontré con una gran dificultad, la falta de documentación relacionada con este tipo de archivos, ya que generalmente ésta se limita a la contenida en su sitio web oficial y en algunos momentos eché de menos poder contrastar ideas o buscar diferentes vías de desarrollo para este tipo de documentos.

Por otra parte, este proyecto me ha permitido aplicar muchos de los conocimientos aprendidos durante la carrera y asentar, todavía más, muchos de ellos.

Creo que el mundo del intercambio de información entre diferentes centros u organizaciones de distintos países es algo fundamental en el campo de la investigación oceanográfica, por este motivo creo el proyecto realizado ha sido de gran interés y los resultados han sido satisfactorios. Estoy convencido de que se seguirán investigando y desarrollando nuevos métodos para facilitar y simplificar estas tareas tanto como sea posible, ya que será beneficioso para todos nosotros.

11. Bibliografía

[1] Cay S. Horstmann, Gary Cornell. "Java 2" (Volumen I/II).

[2] Russ Rew, Glenn Davis, Steve Emmerson, Harvey Davies, Ed Hartnett, and Dennis Heimbigner. "netCDF Documentation" [en línea]. Disponible en: http://www.unidata.ucar.edu/software/netcdf/docs/

[3] John Caron. "NetCD-Java (version 2.2) User's Manual". August 18, 2004.

[4] UNIDATA. "NetCD Java Tutorial". [en línea] Disponible en: http://www.unidata.ucar.edu/software/netcdf-java/tutorial/

[5] UNIDATA. "NetCDF Java Version 4.2 Library and Documentation". [en línea] Disponible en: http://www.unidata.ucar.edu/downloads/netcdf/netcdf-java-4/index.jsp

[6] Brian Eaton, NCAR; Jonathan Gregory, Hadley Centre, UK Met Office; Bob Drach, PCMDI, LLNL; Karl Taylor, PCMDI, LLNL; Steve Hankin, PMEL, NOAA. "NetCDF Climate and Forecast (CF) Metadata Conventions (Version 1.6)".

[7] SeaDataNet. "Metadata Services". [en línea] Disponible en: http://www.seadatanet.org/Metadata/

[8] SeaDataNet. "Data Access". [en línea] Disponible en: http://www.seadatanet.org/Data-Access

[9] SeaDataNet. "Standars & Software". [en línea] Disponible en: http://www.seadatanet.org/Standards-Software

[10] Ocean Data View. "Ocean Data View Getting Started". January 20, 2011. Disponible en: http://odv.awi.de/en/documentation/

[11] Bernardino Casas. "Com documentar un PFC" [en línea]. V 2.1 2007 .Disponible en: http://www.lsi.upc.edu/~bcasas/docencia/pfc/manual_PFC.pdf

12. Apéndice

Como se comentó en el capítulo 9, para realizar el registro del grupo de investigación SARTI-UPC y del observatorio submarino OBSEA en las redes de sistemas de observación era necesario rellenar un formulario, proporcionado por EDIOS, donde se debía introducir información relacionada con la institución y la plataforma.

A continuación se adjunta dicho formulario conforme fue enviado al IEO (Instituto Español Oceangráfico) para realizar el resgistro.

EUROPEAN DIRECTORY OF THE INITIAL OCEAN-OBSERVING SYSTEM (EDIOS)
METADATA INPUT FORM - PART A

PLEASE NOTE MANDATORY FIELDS ARE INDICATED BY AN * AND ARE IN BOLD TYPE

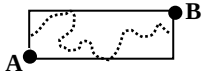
(1) PLATFORM INFORMATION:

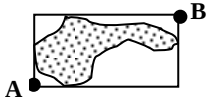
If the observations are collected at one platform then enter the details below. If there are several platforms of the same type, with the same owner, operator and instrumentation then either provide a list of the platform names or use separate forms. Where a repeated cruise is undertaken by different ships, these can count as one platform. However, if possible, please list the names, owners, etc., of all ships.

Name or Identifier of Platform *	OBSEA (Expandable Seafloor Observatory)
Type of Platform *	Cabled Seabed multiparameter Observatory
Platform Owner *	Universitat Politècnica de Catalunya - SARTI
Platform Operator *	Universitat Politècnica de Catalunya - SARTI
Country responsible *	SPAIN

(2) POSITION INFORMATION:

If the observations made from the platform were collected at a single latitude and longitude (example 1 below), use the latitude-1 and longitude-1 columns. If the observations were collected over a range of latitudes (examples 2 and 3), use both latitude and longitude columns and note in the Type column whether this range represents a track or an area. The position information could be for example a series of CTD stations collected along a section, moored buoy positions, continuous plankton recorder tows, or the area covered by a fisheries survey cruise. If this information is already held in digital form, please supply it in its current form (e.g. as ASCII files output from spreadsheets or databases). The Originator's Identifier can be used to indicate a station name, a standard ship route (e.g. SOOP track identifier), or in the case of a mooring, for example, could be the same as the platform identifier. Please also include information about the reference frames and levels.

Examples: 1. Point: **A** • 2. Tracks (or lines): 

3. Area: 

When several instruments are in use on a platform, there needs to be a link between the platform, instrument and variables or parameters measured. The 'Instrument/variable pointer' acts as this link. When completing the Metadata Input Form choose a letter or group of letters as the link, and insert it at the appropriate place in the position, instrument and variable/parameter tables.

Instrument/ variable pointer	*Originator's identifier*	***Latitude-1****	***Longitude-1****	*Latitude-2*	*Longitude-2*	***Type (e.g. point, line or track, area)****	*Min. depth (m)*	*Max. depth (m)*	*Minimum sea floor depth (m)*	*Maximum sea floor depth (m)*	***Start Date****
OBSEA	OBSEA-1	41° 10.927'N	1° 45.151'E			Point	20	20	20	20	May '09

Geographic frame of reference (e.g. WGS84):	
Reference level for depths (e.g. relative to chart datum, mean sea level, instantaneous water surface, sea floor, etc.):	

(3) INSTRUMENT INFORMATION:

Please supply information relating to the instruments used to make the measurements, or attach the information if it is available in digital or printed form. If many instruments are in use, add them in to the EDIOS_MIF_INST file.

Instrument/variable pointer	
Instrument Type*	Cabled Seabed multiparameter Observatory
Instrument Name*	OBSEA
Manufacturer and model	Developed by the SARTI research group from the "Universitat Politècnica de Catalunya"
Manufacture date	May 2009
Description	6 square meters seabed observatory installed at 20 meters depth and 4km offshore the coast of "Vilanova i la Geltrú" (Barcelona). With capacity for 8 ethernet instruments powered at 12 or 48 Volts and 3 amps.
Technical characteristics	At present it is equipped with CTD (temperature, conductivity and depth), hydrophone, video camera, seismometer, acoustic waves and current profiler (AWAC), and surface buoy with weather station.
Further relevant information	
Other attached equipment	Data storage servers at shore station

Instrument/ parameter pointer	CTD1
Instrument Type*	CTD (Conductivity, Temperature, Depth)
Instrument Name*	SeaBird SBE-37 SMP MicroCAT
Manufacturer and model	SeaBird - SBE-37 SMP MicroCAT
Manufacture date	
Description	
Technical characteristics	
Further relevant information	
Other attached equipment	

Instrument/ parameter pointer	HYD
Instrument Type*	Hydrophone
Instrument Name*	Bjørge Naxyx - Ethernet Hydrophone 02345
Manufacturer and model	Bjørge Naxyx - Ethernet Hydrophone 02345
Manufacture date	
Description	
Technical characteristics	
Further relevant information	
Other attached equipment	

Instrument/ parameter pointer	CAMERA
Instrument Type*	TV-camera
Instrument Name*	Underwater IP Camera Sony SNC-RZ25N
Manufacturer and model	
Manufacture date	
Description	

Technical characteristics	
Further relevant information	
Other attached equipment	

(4) VARIABLE/PARAMETER INFORMATION:
List the variables or parameters measured (e.g. temperature, salinity, chlorophyll a, phytoplankton, dissolved oxygen, pH, particulate carbon, silicate, etc), including where possible, an estimate of their accuracy. Any standard real-time or delayed-mode validation schemes should also be noted.

Instrument/ parameter pointer	***Variable/Parameter****	*Accuracy*	*Real-time data validation?*	*Delayed-mode validation?*	*Sampling frequency (e.g. 10 minute, hourly, daily, annually,3 hours before high water, etc.)*
CTD	Temperature	± 0.002º C			10 seconds
	Conductivity	±0.0003 S/m			10 seconds
	Depth				
	Salinity				10 seconds
	Sound Velocity				10 seconds
HYD	Sound				
CAM	Images				

(5) QUALITY MANAGEMENT SYSTEM
Complete for each platform, group of platforms, instrument or group of instruments as appropriate

Is there a Quality Management System (QMS) followed by the organisation responsible for the observations?*	*Yes*	*No*

If yes, is the institution accredited or certified? *		***No***
If yes, indicate *accreditation scheme (e.g. ISO9000, QUASIMEME)*		
If no, please note which of the following quality assurance procedures are applied (or attach appropriate documentation)		
How frequently are sensors calibrated?	*Annually*	
Are sensors calibrated prior to and after the measurement period?	*Yes*	
Is calibration performed by the manufacturer or by the data generating institution?	*Manufacture*	
To what accuracy are the sensors calibrated?		
Are there sensor intercalibrations (regular or not)?	*Not yet.*	
Are there back-up sensors?	*Yes*	
For automatic recording stations: Frequency of comparative in situ measurements (for each variable).	*Sporadically salinity measurements comparative*	
Is there an instant data validation based on neighbouring stations?	*NO*	
Are data gaps filled by derived values (e.g. after multi-linear regression)?	*NO*	
Are standard procedures applied for the processing of the (raw) data (e.g. removal of spikes, comparison with existing climatologies etc.)?	*NO*	
Are there additional quality control checks, such as consistency checks, range of validity, date, geographical position (typical NODC control procedures)?	*YES*	
In chemistry: Are there intercalibration exercises? Proficiency testing? Internal quality assurance procedures, e.g. control charts, blank analyses? Name E(uropean) N(orm,) NODC?	*NO*	
Are the high quality data and the corresponding metadata safe-guarded in a central data bank (NODC, WDC)	*NO*	
Include details of any other relevant procedures in use		

(6) OBSERVING PROGRAMME INFORMATION

Observing programme name (e.g. UK tide gauge network, UK MAWS (met. Buoy) network, CTD section, Swedish coastal monitoring stations)*
Description of programme
Measurments are not included in any Observing program.
National/International project name(s) (e.g. Argo, GODAE, MedGOOS)

PROGRAMME STATUS: ***Please note future commitment to the observing programme, adding dates where appropriate ****		
An active measurement programme with no planned end	Yes	No
An active measurement programme with a planned end (include end date if known)		
An active measurement programme with a planned end and planned repeat of program (include dates if known)		
An non-active measurement programme with planned repeat of the program (include dates if known)		
Status unknown		

ACCESS TO THE DATA: *Include information relating to the conditions and protocols for access to the measurements or attach the relevant information (e.g. unrestricted access, access for registered users, access for subscribed users, access restricted, confidential, available for scientific research only, available under EuroGOOS data policy).*
• *Access to data held at the centre responsible for the observations*
access for registered users
• *Access to real-time observations*
access for registered users
• *Access to the archived data set from the observing programme*
access for registered users
Web-site for access to real-time data
www.obsea.es
Web site for access to archived data
www.obsea.es

EUROPEAN DIRECTORY OF THE INITIAL OCEAN-OBSERVING SYSTEM (EDIOS)
METADATA INPUT FORM - PART B

PLEASE NOTE MANDATORY FIELDS ARE INDICATED BY AN * AND ARE IN BOLD TYPE

Complete the sections below to provide contact information. Section 1 (Responsible organisation) should always be completed for each platform or group of platforms.

(1) RESPONSIBLE ORGANISATION CONTACT INFORMATION

NAME OF ORGANISATION*	Universitat Politècnica de Catalunya. SARTI Research Group
CONTACT TITLE (e.g. position/title of post for contact person)*	Dr. Joaquin del Rio Fernandez
TELEPHONE	+34938967200
FAX	
E-MAIL	Joaquin.del.rio@upc.edu
WEB-SITE	http://www.cdsarti.org
BUILDING (building name or number)*	Centre Tecnològic de Vilanova i la Geltrú
STREET*	Rambla Exposició 24 Edifici, C
TOWN/CITY*	Vilanova i la Geltrú
REGION (e.g. county/state/region)*	Barcelona (Catalunya)
***ZIP/POSTCODE **	08800
COUNTRY*	Spain

BRIEF DESCRIPTION OF THE ORGANISATION
Research group devoted mainly to marine instrumentation research and development.

(2) REAL-TIME DATA CONTACT INFORMATION

NAME OF ORGANISATION	Universitat Politècnica de Catalunya. SARTI Research Group
CONTACT TITLE (e.g. position/title of post for contact person)	Dr. Joaquin del Rio Fernandez
TELEPHONE	+34938967200
FAX	
E-MAIL	Joaquin.del.rio@upc.edu

WEB-SITE	http://www.cdsarti.org
BUILDING (building name or number)	Centre Tecnològic de Vilanova i la Geltrú
STREET	Rambla Exposició 24 Edifici, C
TOWN/CITY	Vilanova i la Geltrú
REGION (e.g. county/state/region)	Barcelona (Catalunya)
ZIP/POSTCODE	08800
COUNTRY	Spain

BRIEF DESCRIPTION OF THE ORGANISATION

(3) DATA ARCHIVE CENTRE CONTACT INFORMATION

NAME OF ORGANISATION	Universitat Politècnica de Catalunya. SARTI Research Group
CONTACT TITLE (e.g. position/title of post for contact person)	Dr. Joaquin del Rio Fernandez
TELEPHONE	+34938967200
FAX	
E-MAIL	Joaquin.del.rio@upc.edu
WEB-SITE	http://www.cdsarti.org
BUILDING (building name or number)	Centre Tecnològic de Vilanova i la Geltrú
STREET	Rambla Exposició 24 Edifici, C
TOWN/CITY	Vilanova i la Geltrú
REGION (e.g. county/state/region)	Barcelona (Catalunya)
ZIP/POSTCODE	08800
COUNTRY	Spain

BRIEF DESCRIPTION OF THE ORGANISATION

EDIOS Metadata Input Form

COMPLETED BY: Joaquin del Rio

DATE: Juny 15th 2012