Inicializar Formulario | Guardar Portadas TFM

Rellenar este documento, guardar y agregar al principio del documento del TFM como portadas.

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en
Tecnologías de Telecomunicación

Trabajo Fin de Máster

SmartPort: Aplicación para la visualización de grandes volúmenes de datos (Big Data) proveniente de sensores conectados a internet (IoT).

Autor: **Pablo Fernández Moniz**
Tutor(es): **Ángel Plaza de la Hoz**
Jose Pablo Suárez Rivero
Fecha: **Julio 2015**

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en
Tecnologías de Telecomunicación

Trabajo Fin de Máster

SmartPort: Aplicación para la visualización de grandes volúmenes de datos (Big Data) proveniente de sensores conectados a internet (IoT).

HOJA DE FIRMAS

Alumno/a: Pablo Fernández Moniz Fdo.:

Tutor/a: Ángel Plaza de la Hoz Fdo.:

Tutor/a: Jose Pablo Suárez Rivero Fdo.:

Fecha: Julio 2015

IUMA
t +34 928 451 086 | iuma@iuma.ulpgc.es
f +34 928 451 083 | www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

Versión: 2.0 – 14 jun 2011

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en
Tecnologías de Telecomunicación

Trabajo Fin de Máster

SmartPort: Aplicación para la visualización de grandes volúmenes de datos (Big Data) proveniente de sensores conectados a internet (IoT).

HOJA DE EVALUACIÓN

Calificación: .....

Presidente: Fdo.:

Secretario: Fdo.:

Vocal: Fdo.:

Fecha: **Julio 2015**

IUMA
t +34 928 451 086
f +34 928 451 083
iuma@iuma.ulpgc.es
www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

Versión: 2.0 – 14 jun 2011

SmartPort: Aplicación para la visualización de grandes volúmenes de datos (*Big Data*) proveniente de sensores conectados a internet (IoT).

Alumno: Pablo Fernández Moniz

IUMA | Instituto Universitario de Microelectrónica Aplicada

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

SmartPort: Aplicación para la visualización de grandes volúmenes de datos (*Big Data*) proveniente de sensores conectados a internet (IoT).

Máster IUMA
Alumno: Pablo Fernández Moniz

Índice

1.Introducción y Objetivos

1.1 Resumen

SmartPort se presenta como una aplicación que se ha desarrollado en el proyecto FIWARE financiado por el séptimo programa marco.

El producto resultante ha sido una herramienta para la gestión de datos provenientes de diferentes instrumentos conectados a Internet o también conocido como Internet of Things (IoT).

Dichos datos se capturan mediante sensores geolocalizados en el territorio que pueden ser fijos, tales como estaciones meteorológicas y boyas, o de ubicación variable, como pueden ser los barcos con su Sistema de Identificación Automática (SIA).

El backend de la aplicación se ha desarrollado con el ecosistema de FIWARE; un aglomerado de tecnologías modulares, denominadas Generic Enablers que usan, entre otras soluciones, MongoDB o Hadoop.

Esto nos permite la gestión de información en vivo y de grandes volúmenes de datos (*Big Data*). Además, gracias a las librerías de visualización de datos geográficos en tres dimensiones, en este caso con Glob3Mobile, hemos sido capaces de mostrar elementos del puerto usando modelos tridimensionales que aportan un valor añadido a la hora de representar la información geográfica.

Con la suma de todo esto obtenemos una aplicación web enriquecida en la que podemos ver dónde están los sensores en el territorio, los últimos datos que nos han proporcionado y un histórico de los últimos años mediante gráficas exportables. No sólo nos da la posibilidad de observar la información; tiende a convertirse en un cuadro de mando gracias a funcionalidades como un sistema de alertas que nos avisa cuando un sensor toma un valor determinado, y de este modo, poder tomar decisiones a la hora de gestionar el puerto.

PALABRAS CLAVE: SIG, BIG DATA, SMART PORT, SMART CITY, FIWARE, GLOB3MOBILE, IOT, OPEN DATA, CUADRO DE MANDO GEOGRÁFICO.

1.2 Abstract

SmartPort is an application that has been developed by the project FIWARE funded under the seventh framework program.

The resulting product is a tool for managing data from different instruments connected to the Internet or also known as the Internet of Things (IoT).

Data is captured by geolocated sensors in the territory that can be in a fixed location, such as weather stations and buoys, or with variable location, such as ships with Automatic Identification System (AIS).

The backend of the application is developed under FIWARE ecosystem (a cluster of modular technologies, called Generic Enablers) using, among other solutions, MongoDB and Hadoop.

This allows us to manage live information and large volumes of data (Big Data). Thanks to 3D geovisualization libraries, in this case Glob3Mobile, we were able to show elements of the port using three-dimensional models that add value in representing geographic information.

Summing of all this, we get a rich web application in which we can see where the sensors are in the territory, recent data taken by them and historical data through exportables graphs. It does not only give us the opportunity to observe the information, its goal is to become a scorecard thanks to features such as an alert system that warns you when a sensor takes a certain value and, thus, allowing us to manage the port.

KEYWORDS: GIS, BIG DATA, SMART PORT, SMART CITY, FIWARE, GLOB3MOBILE, IOT, OPEN DATA, SCORECARD.

1.3 Objetivos del trabajo fin de máster.

Mediante un convenio con la Autoridad Portuaria de Las Palmas de G.C. surgió SmartPort, a partir de este proyecto, realicé un caso práctico que dio lugar al TFM, cuyos objetivos son los siguientes:

1. Manejar y gestionar los datos recientes de los sensores mediante servicios REST

2. Almacenar y consultar el histórico de datos mediante un sistema gestor de *Big Data*

3. Mostrar la información geolocalizada en un visor en 3D

4. Desarrollar un módulo de alertas para la toma de decisiones

5. Implementación en un servidor para mostrar la aplicación sobre web

1.4 Producción científica

Como resultado de este trabajo fin de máster, un grupo de investigadores que incluyen al CTIM y al IUMA hemos desarrollado cierta actividad de difusión científica, que se resumen en las siguientes aportaciones:

Jose Pablo Suárez Rivero, Agustín Trujillo Pino, Conrado Domínguez Trujillo, Pablo Fernandez Moniz, Jaisiel Santana Almeida, Alejandro Sánchez Medina, Sebastián Ortega Trujillo, Jose Miguel Santana Nuñez. SmartPort: sistema para la visualización y gestión de información del Puerto de la Luz de las Palmas de G.C., Jornadas de SIG libre de Girona, 2015

Pablo Fernández, José P. Suárez, Agustín Trujillo, Conrado Domínguez, José M. Santana. Managing and 3D Visualization of Real-time Big Geo-referenced Data from Las Palmas Port through a Flexible Open Source Computer Architecture, GISTAM congress 2015.

Unido a estos artículos actualmente estamos redactando una publicación para una revista de impacto.

Además el trabajo sido precursor de nuevas líneas de trabajo relacionadas con aplicaciones web, móviles y *Big Data* de un equipo de trabajo de la división MAGIC del IUMA.

2. Estado del arte

2.1 Sistemas de gestión de bases de datos relacionales

Un Sistema Gestor de Bases de Datos Relacional (SGBDR) es el es el modelo más utilizado en la actualidad para implementar bases de datos modelables.

Permiten establecer interconexiones (relaciones) entre datos y a través de dichas conexiones relacionar los datos de varias tablas, de ahí proviene su nombre: "Modelo Relacional".

Los principios de las Bases de Datos Relacionales, fueron postuladas en 1970 por Edgar Frank Codd, con el artículo "*A Relational Model of Data for Large Share Data Banks*" [1].

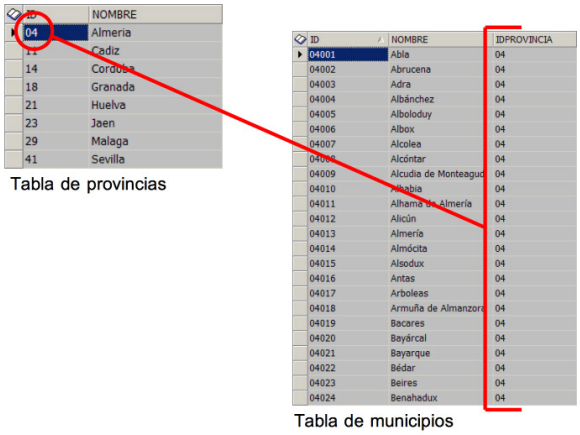


Figura 1. Relaciones entre tablas en un modelo Entidad-Relación

Dentro de los SGBDR es importante conocer si implementa una característica muy importante: ACID (Atomicity, Consistency, Isolation and Durability).

En bases de datos se denomina ACID a las características de los parámetros que permiten clasificar las transacciones de los sistemas de gestión de bases de datos.

Cuando se dice que es ACID compliant se indica que cumple una serie de parámetro y éste permite realizar transacciones.

A continuación definimos cada uno de estos items:

Atomicidad: Si una operación consiste en una serie de pasos, todos ellos ocurren o ninguno, es decir, las transacciones son completas.

Consistencia: Es la propiedad que asegura que sólo se empieza aquello que se puede acabar.

Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de Integridad de la base de datos. La propiedad de consistencia sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro también válido. "La Integridad de la Base de Datos nos permite asegurar que los datos son exactos y consistentes, es decir que estén siempre intactos, sean siempre los esperados y que de ninguna manera cambien ni se deformen. De esta manera podemos garantizar que la información que se presenta al usuario será siempre la misma."

Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error. Esta propiedad define cómo y cuándo los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes. El aislamiento puede alcanzarse en distintos niveles, siendo el parámetro esencial a la hora de seleccionar SGBD.

Durabilidad: Persistencia. Es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema y que de esta forma los datos sobrevivan de alguna manera.

Cumpliendo estos 4 requisitos un sistema gestor de bases de datos puede ser considerado ACID Compliant.

Otras características que nos interesan de un SGBDR es el tamaño máximo de base de datos y de tabla permitido, su interfaz y si dispone de extensión para el trabajo de datos geográficos.

MySQL

MySQL [2] es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

Se encuentra bajo el paraguas de la empresa MySQL AB, que desde enero de 2008 es una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009.

Por un lado se ofrece bajo la GNU GPL, en el caso de la versión Community Edition. Para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como sucede en Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

MySQL es muy utilizado en aplicaciones web, por lo que encontramos relacionado con este SGBDR muchos proyectos del mundo web, tales como Wordpress y Drupal.

MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero esto puede provocar problemas de integridad en entornos de alta concurrencia en la modificación.

También ha tomado mucha fama gracias a las implementaciones XAMPP, el nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl.

Existe un proyecto MySQL Spatial para trabajar con información georreferenciada.

PostgreSQL

PostgreSQL [3] es un Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia BSD.

Cuenta con más de 15 años de desarrollo activo, iniciado en la Universidad de Berkeley en 1982, y una arquitectura probada que se ha ganado una sólida reputación por su fiabilidad, integridad de datos y corrección.

Es una base de datos de clase empresarial. PostgreSQL cuenta con características sofisticadas como Multi-Version Concurrency Control (MVCC), punto en el tiempo de recuperación, tablespaces, replicación asincrónica, transacciones anidadas (puntos de retorno) etc.

PosgreSQL se presenta como el SGBDR completamente libre más ponente del mercado. Es muy útil para proyectos centrados en la gestión y manejo de datos.

Contiene un módulo de análisis espacial: PostGIS

Oracle

Oracle Database [4] es un SGBDR privativo desarrollado por Oracle Corporation.

Oracle Database se considera como uno de los sistemas de bases de datos propietarios más completos, destacando:

- soporte de transacciones.

- estabilidad.

- escalabilidad.

- soporte multiplataforma.

Oracle se presenta como el SGBDR referencia para grandes bases de datos. Grandes clientes, como la seguridad social, Policía Nacional etc. usa Oracle como gestor de bases de datos.

Es criticado por sus abusivas licencias y complejidad del sistema con un sistema de negocio adicional al licenciamiento creando certificaciones y formación impartido por la compañía.

Mediante sus módulos Locator y Spatial ofrece la posibilidad de realizar trabajos con geometrías espaciales.

Microsoft SQL Server

Microsoft SQL Server [5] es un SGBDR propietario desarrollado por la empresa Microsoft.

Se caracteriza por correr sólo bajo Windows, lo cual puede ser considerado como poco recomendable para servidores que proporcionen servicios estables.

Microsoft SQL server tiene las siguientes características:

- Soporte de transacciones.

- Capacidad de usar procedimientos almacenados.

- Incluye también un entorno gráfico de administración, que permite el uso de comandos DDL (Data Definition Language) y DML (Data Manipulation Language) gráficamente.

- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.

- Además permite administrar información de otros servidores de datos.

A partir de la versión 2008 dispone de capacidad de trabajo con datos geográficos

Microsoft Access

Microsft Access [6] es un programa desarrollado por Microsoft que emula el funcionamiento de un SGBDR.

Carece de principios básicos de trabajo de un SGBDR de producción tales como la multiconcurrencia o implementación ACID.

Es un programa muy cómodo para trabajar con pequeños volúmenes de datos (inferiores a 2GB), a nivel de

ofimática o para tener pequeñas bases de datos.

No es transaccional y tampoco presenta posibilidad de trabajar con información geográfica.

Presenta la opción de programas pequeñas funciones mediante VBA (Visual Basic for Applications).

Comparativa entre los SGBDR expuestos

En resumen no podemos que existe un SGBDR mejor a otro, ya que según el propósito que se tenga será necesario decantarse por una solución u otra, incluso en algunos casos, hay que hacer una combinación entre diferentes SGBDR

Tabla 1 comparativa entre SGBDR

	Oracle	PostgreSQL	MySQL	MS SQL Server	MS Access
ACID	Si	Si	Si	Si	No
Tamaño max. de DB	Ilimitado	Ilimitado	Ilimitado	Ilimitado	2 GB
Tamaño max. de tabla	Ilimitado	32 TB	2 GB (Win FAT32), 16 TB (resto)	524 TB	2 GB
Interfaz	GUI (propio, SQL Developer, TOAD), consola SQL (SQL*PLUS)	GUI (Kpogre, PgAdmin3), consola SQL (PSQL)	GUI(phpMyAdmin, MySQL Administrator, SQLyog), consola SQL (MYSQL)	GUI (propio), consola SQL (SQLCMD)	GUI (propio)
Notas	Líder mundial software propietario. CaracterÌsticas muy avanzadas. Arquitectura cliente-servidor.	Colíder mundial FOSS. Características muy avanzadas. Arquitectura cliente-servidor.	Colíder mundial FOSS (aplicaciones web). Caractertícas muy avanzadas. Muy popular gracias a despliegues LAMP. Arquitectura cliente-servidor.	De Microsoft. Implantación limitada. Características muy avanzadas. Arquitectura cliente-servidor.	De Microsoft. Base de datos no transaccional. No plantеable para aplicaciones crÌticas. Características limitadas. No multiconcurrente.
Extensión geográfica	Si. Oracle Locator (por defecto), Oracle Spatial (módulo adicional).	Si. PostGIS, basada en GEOS y Proj4.	Si. MySQL Spatial (aún en testing).	Si, a partir de la versión 2008.	No (ni se le espera).

2.2 Sistemas tipo NoSQL para la gestión de datos

En informática, NoSQL (Not only SQL) es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales.

Su nombre pretende destacar que usa otros lenguajes de consulta además de SQL.

Algunas de las principales carácterísticas son:

- Los datos almacenados suelen usar estructuras fijas como tablas.

- No pretende usar los JOIN de manera tan usual como los SGBDR.

- No garantizan completamente la aplicación de ACID.

-Habitualmente escalan bien horizontalmente.

A menudo, las bases de datos NoSQL se clasifican según su forma de almacenar los datos, y comprenden categorías como clave-valor, las implementaciones de BigTable, bases de datos documentales, y bases de datos orientadas a grafos como veremos a continuación.

Los sistemas de bases de datos NoSQL crecieron con las principales compañías de Internet, como Google, Amazon, Twitter y Facebook. Estas empresas tenían que enfrentarse a desafíos con el tratamiento de datos que las tradicionales SGBDR no solucionaban de manera eficiente.

Con el crecimiento de la web en tiempo real existía una necesidad de proporcionar información procesada a partir de grandes volúmenes de datos que tenían unas estructuras horizontales más o menos similares.

Estas compañías se dieron cuenta de que el rendimiento y sus propiedades de tiempo real eran más importantes que la coherencia, en la que las bases de datos relacionales tradicionales dedicaban una gran cantidad de tiempo de proceso.

En ese sentido, a menudo, las bases de datos NoSQL están altamente optimizadas para las operaciones recuperar y agregar, y a menudo no ofrecen mucho más que la funcionalidad de almacenar los registros.

La pérdida de flexibilidad en tiempo de ejecución, comparado con los sistemas SQL clásicos, se ve compensada por ganancias significativas en escalabilidad y rendimiento cuando se trata con ciertos modelos de datos

Cabe destacar, que cuando hablamos de sistemas NoSQL no hablamos de un sólo programa como en los SGBDR, habitualmente es un stack o conjunto de tecnologías.

A continuación tratamos las más importantes:

Bases de datos NoSQL

MongoDB

MongoDB [7] es una base de datos de almacenamiento documental desarrollado por MongoDB Inc. y lanzado por primera vez en 2009.

Se distribuye bajo la licencia GNU AGPLv3 y su último lanzamiento fue la versión 3.0.3 (mayo 2015).

CouchDB

CouchDB [8] es una base de datos de almacenamiento documental desarrollado por la Apache Software Foundation y su primera versión fue lanzada en 2005.

Se distribuye bajo la licencia Apache v2.0 y su último lanzamiento fue version1.6.1 (en septiembre de 2014).

DynamoDB

DynamoDB [9] es un base de datos de almacenamiento documental y de clave-valor desarrollado por Amazon y lanzado por primera vez en 2012.

Se publica bajo licencia propietaria. Sólo está disponible como un servicio en la nube.

HBase

HBase [10] es una base de datos de almacenamiento de grandes volúmenes de datos distribuidos desarrollado por la Apache Software Foundation y la primera versión fue lanzada en 2008.

Se distribuye bajo la licencia Apache v2.0 y su último lanzamiento fue la versión 1.1.01 (mayo 2015).

Cassandra

Cassandra [11] es una base de datos de almacén de la columna ancha desarrollado por la Apache Software Foundation y la primera versión fue lanzada en 2008.

Se distribuye bajo la licencia Apache v2.0 y su último lanzamiento fue la versión 2.1.6 (en junio de 2015.

Accumulo

Accumulo [12] es una base de datos basado en Google Big Table, y Hadoop. Desarrollado por la Apache Software Foundation y la primera versión fue lanzada en 2012.

Se distribuye bajo la licencia Apache v2.0 y su último lanzamiento fue la versión 1.6.1 (en octubre de 2014).

Redis

Redis [12] es una base de datos de almacenamiento clave-valor en caché y almacenamiento desarrollado por Salvatore Sanfilippo (actualmente patrocinado por Pivotal) y fue lanzado por primera vez en 2009.

Se distribuye bajo la licencia BSD de 3 Cláusula y su último lanzamiento fue la versión 3.0.2 (en junio de 2015).

Riak

Riak [13] es una base de datos de tipo clave-valor desarrollado por Basho Technologies y la primera versión fué lanzada el 2009.

Se distribuye bajo la licencia Apache v2.0 y su último lanzamiento fue la versión 2.1.0 (en abril de 2015).

Neo4j

Neo4j [14] es una base de datos de almacenamiento orientada grafos desarrollado por Neo Technology y la primera versión fue lanzada el 2007.

Se distribuye bajo la licencia GPLv3 y su último lanzamiento fue la versión 2.2.0 (marzo 2015).

OrientDB

OrientDB [15] es una base de datos orientada a grafos de segunda generación, desarrollada por Orient Technologies Ltd y lanzado por primera vez en 2010.

Se distribuye bajo la licencia Apache v2.0 y su último lanzamiento fue la versión 2.0.10 (mayo 2015).

Aerospike

Aerospike [16] es una base de datos del almacén de claves-valor desarrollado por Aerospike y la primera versión fue lanzada el 2012.

Se distribuye bajo las licencias GNU v3.0 AGPL y su último lanzamiento fue la versión 3.5.3 (en febrero de 2015).

Hypertable

Hypertable [17] es una base de datos de grandes volúmenes de datos desarrollada por Hypertable Inc. y fue lanzado por primera vez en 2009.

Se distribuye bajo la licencia GNUv3 y su último lanzamiento fue la versión 0.9.7.2 (en abril de 2013).

Ecosistema Hadoop

Hadoop [18] es un framework especializado en el procesamiento distribuido de datos. Su arquitectura está pensada para ofrecer escalado en horizontal, empleando hardware con características no demasiado exigentes.

Hadoop ofrece dos funcionalidades básicas: un sistema de ficheros distribuido llamado Hadoop Distributed File Sytem (HDFS) [19], inspirado en Google File System (GFS)[20]; y un paradigma de procesamiento distribuido conocido como MR (MapReduce) [21].

Tomando como base Hadoop, podemos encontrar una gran cantidad de herramientas que pretenden facilitar diferentes tareas.

El proyecto Apache ofrece varias de ellas, las cuales conforman el nombrado ecosistema Hadoop:

- Ambari [22]: Herramienta web para a administración de clusters de Hadoop (provisionamiento, manejo y monitorización).

- Avro [23]: Herramienta para la serialización de datos.

- Hbase[24]: Base de datos no relacional tabular que permite realizar operaciones de escritura/lectura en HDFS.

- Hive [25]: Herramienta para ofrecer sintaxis de consultas SQLs para tareas MapReduce.

- Mahout [26]: Librería para minería de datos y aprendizaje automático.

- Pig [27]: Herramienta para ofrecer sintaxis de flujo de datos para tareas MapReduce.

- Zookeeper [28]: Servicio de coordinación de aplicaciones distribuidas.

Posteriormente encontramos una serie de implementaciones comerciales, que viene a facilitar la instalación y las denominadas variables del entorno a la hora de implementar un stack de trabajo.

Hortonworks [29]

MapR [30]

Cloudera [31]

Criterios de comparación entre los diferentes gestores de datos

Según el modelo

- Tipo Documental

Las bases de datos documentales o sistemas de gestión de base de datos documental son bases de datos que pretenden organizar los datos en una forma sin esquema.

Los registros almacenados se denominan documentos y compilar un conjunto de atributos o columnas.

Cada atributo puede tener un número variable de valores (el caso de una matriz). Lo que es importante es que los documentos de la misma categoría pueden tener un número diferente de atributos, los atributos pueden ser de diferentes tipos y los valores pueden ser documentos mismos, en el caso de documentos anidados.

Todas estas características componen la definición de una gestión de datos sin esquema.

- Tipo Clave-valor

Los sistemas de gestión de bases de datos clave-valor almacenan sólo tuplas almacenamiento con dos campos nombrados clave y valor.

En estos sistemas, un valor sólo puede ser recuperada cuando se conoce la clave relacionada.

- Tipo de columnas anchas

Los sistemas de gestión de bases de datos almacena las columnas o grabar amplias columnas también organizan los datos de una manera sin esquema.

En este caso, su aplicación consta en los registros, las columnas, con número variable de entradas, siendo normalmente un conjunto muy grande.

Se les puede ver como almacenamiento clave-valor en dos dimensiones .

- Orientada a grafos.

Los sistemas de gestión de bases de datos almacena orientada a grafos contienen datos como nodos y bordes, siendo los bordes de las relaciones entre los nodos.

Esta estructura hace más fácil el cálculo de propiedades de la gráfica como el número de saltos entre los nodos.

Características técnicas

Almacenamiento de datos:

La memoria volátil: La base de datos está optimizado para hacer uso de la memoria RAM para el almacenamiento.

Sistema de archivos: La base de datos persiste datos para el sistema de archivos del sistema operativo.

SSD (solid-state drive) La base de datos está optimizado para trabajar Unidos discos sólidos en lugar de hacer girar el disco físicamente.

HDFS: La base de datos se construye en la parte superior del sistema de archivos distribuido Hadoop y persiste los datos en él.

Bitcask [32]: Es una aplicación Erlang para la clave de almacenamiento / valor en el disco. Se basa en los sistemas de archivos de registro estructurado

LevelDB [33]: Un proyecto de Google para la clave de almacenamiento / valor en el disco. Se basa en archivos de registro y ordenadas según tablas de claves.

Lenguaje de consulta

JavaScript: La base de datos cuenta con un lenguaje de consulta que la sintaxis se basa en expresiones de JavaScript.

Llamadas API: La base de datos no proporciona una sintaxis concreta para consultar datos. En su lugar, proporciona una API de consulta que se implementa en varios lenguajes de programación.

Comandos HBase Shell: El programa de shell para HBase proporciona comandos personalizados que hacer consultas básicas.

CQL: El Lenguaje de Consulta Cassandra es una variación de SQL inspirado por el proyecto Apache Cassandra.

Lucene: La base de datos cuenta con un lenguaje de consulta que la sintaxis se basa en expresiones Lucene.

Cypher: La base de datos proporciona Cypher Lenguaje de consulta para hacer consultas de datos. Está inspirado en la sintaxis SQL, pero está orientado hacia la representación gráfica.

Gremlin: La base de datos proporciona el lenguaje Gremlin consulta para hacer consultas de datos. Está orientado hacia la representación gráfica.

SQL: La base de datos implementa una variante de SQL para su lenguaje de implementación.

Lua UDF: La base de datos proporciona la consulta por Definido por el usuario Función codificado en el lenguaje de programación Lua.

HQL: El lenguaje de consulta Hypertable es una variación de SQL inspirado para el proyecto Hypertable.

Comunicación

Destaca el tipo de comunicación de la base de datos, que trata el protocolo a nivel de aplicación.

Actualizaciones de entrada condicionales

Si la base de datos permite actualizarse de forma nativa para basándose en consultas condicionales.

Por ejemplo: una columna c se cambie a verdadero cuando la columna b sea mayor de 10

Lenguaje de implementación

Lenguaje de programación en el que se implementa la base de datos.

Integridad

ACID: Atomicidad, consistencia, aislamiento y durabilidad. Común en SGBDR.

BASE: Representa el siguiente conjunto de propiedades:

Basically available : el sistema siempre responde, incluso si la respuesta indica que los datos no pudieron ser recuperados debido a problemas o actualizaciones).

Soft-state (el sistema podría estar cambiando, incluso si se emiten ningún comando, ya que podría estar convergiendo en una constante estado)

Consistencia eventual (el sistema convergerá a un estado coherente algún tiempo después de la última entrada recibida).

MVCC: Multiversion concurrency control.

En general, los lectores de la base de datos son capaces de ver las instantáneas de los datos mientras se está leyendo.

Cuando el proceso que escribe termina la actualización de datos, el estado anterior está marcado como obsoleto y el lector es capaz de saber que no hay datos actuales, pero todavía son capaces de acceder a él.

Transacciones

Las transacciones son un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.

Una base de datos se dice transaccional, si es capaz de mantener la integridad de los datos, haciendo que estas transacciones no puedan finalizar en un estado intermedio. Cuando por alguna causa el sistema debe cancelar la transacción, empieza a deshacer las órdenes ejecutadas hasta dejar la base de datos en su estado inicial (llamado punto de integridad), como si la orden de la transacción nunca se hubiese realizado.

Integridad referencial

Si la base de datos proporciona mecanismos para hacer referencia a los valores de los campos de otra entidad (documento, fila, par clave-valor)

Control de versiones

Si la base de datos proporciona funciones para hacer frente a versiones de datos en cada disco.

Indexación

- Índices secundarios

Si se pueden definir varios índices en las entidades, además de las claves principales, a fin de optimizar las búsquedas

- Índices compuestos

Si somos capaces de utilizar varias columnas, parejas o campos para definir un índice

- Índices Geoespaciales

Si se permite la base de datos para la definición y el uso de índices geoespaciales

Soporte de grafos

Si la base de datos capaz de proporcionar consultas orientadas a grafos.

Distribución

- Escalabilidad horizontal

Si el sistema de base de datos capaz de proporcionar escalabilidad añadiendo más nodos a la arquitectura

- Replicación

Master-Slave: esquema maestro-esclavo , la replicación de datos ocurre desde un servidor principal / nodo llamado el maestro a cada esclavos. Los datos no se puede sincronizar en el revés. Algunos sistemas definen dos modos para esta replicación suceda:

Continuo: Cada vez que un cambio se produce en un maestro, que se replica en los esclavos tan pronto como sea posible.

One-shot: La replicación se activa mediante el envío de un comando de replicación para el sistema.

Conjuntos de réplica: conjuntos de réplica son grupos de nodos que mantienen el mismo conjunto de datos. La replicación de los datos dentro del conjunto sucede de una manera maestro-esclavo, pero cuando el maestro no está disponible un nuevo maestro es elegido a través de un sistema de dinámico de elección.

Maestro-Maestro: La replicación de datos pasa de un maestro a un nodo esclavo. La diferencia es que ahora el esclavo puede comportarse como un maestro.

Cíclico: El maestro y los nodos esclavos pueden cambiar los roles y los flujos de datos van en ambos sentidos.

Síncrono: En la replicación sincrónica se puede configurar la replicación a cualquier nodo del clúster de replicación definido.

La principal diferencia aquí es que un comando de escritura no se reconoce para tener éxito hasta que todas las réplicas confirman que los datos se han propagado.

Sharding

El concepto de sharding refiere a particionamiento horizontal de un almacenamiento de base de datos.

Por lo tanto, un fragmento se define como una partición horizontal de los datos que se almacena en un nodo particular de la base de datos. El concepto de particionamiento horizontal se refiere a la división del conjunto de datos por filas en lugar de columnas o atributos.

Arquitectura sin nada compartido

Una arquitectura de nada compartido es el tipo de arquitectura distribuida en la que cada nodo es independiente y autosuficiente.

Por lo tanto, el sistema no almacena ningún tipo de información de estado centralizado a fin de coordinar sus nodos. Las ventajas de este tipo de sistemas están evitando los puntos únicos de fallos, la posibilidad de añadir capacidades de autocuración y la posibilidad de realizar actualizaciones sin interrupciones.

APIS / Clientes

Aquí nos mostrará si la base de datos tiene una API con apoyo oficial o un cliente.

Tabla 2 Características generales de sistemas NoSQL

DB	MongoDB	Cou-chDB	DynamoDB	HBase	Cassandra	Accumulo	Redis	Riak	Neo4j	OrientDB	Aeros-pike	Hyper-table
Modelo	Documento	Docu-mento	Documento Clave-valor	Columna	Columna	Columna	Clave-valor	Clave-valor	Grafos	Documen-to Clave-valor Grafo	Clave-valor	Colum-na
Lenguaje	C++	Erlang	Java	Java	Java	Java	C C++	Erlang	Java	Java	C	C++
Almacenamiento	Volatile me-mory File System	Volatile memory File Sys-tem	File Sytem (SSD)	HDFS	File system	HDFS	Volatile me-mory File System	Volatile memory Bitcask LevelDB	Volatile memory File Sys-tem	Volatile memory File System	Volatile memory File system (SSD)	HDFS
Lenguaje de con-sulta	JavaScript	JavaS-cript	API Calls	API Calls HBase shell com-mands	API calls CQL	API calls	API calls	Lucene	Cypher-QL Gremlin	API Calls SQL	Lua UDF	API calls HQL
Comunicación	Custom Binary (BSON)	HTTP/ REST	HTTP/REST	HTTP/ REST Thrift	Binary CQL Thrift	Thrift	Telnet-like Binary	HTTP/ REST Binary	HTTP/ REST	HTTP/ REST Binary	Propie-tary	Thrift
Entrada de actuali-zaciones	Si	Si	Si	Si	No	Si	No	No	No	Si	Si	Si
MapReduce	Si	Si	Si	Si	Si	Si	No	Si	No	Si	Si	Si

Tabla 3 Características de control en sistemas NoSQL

DB	Mongo-DB	Cou-chDB	Dynamo-DB	HBase	Cassan-dra	Accumu-lo	Redis	Riak	Neo4j	OrientDB	Aerospike	Hyperta-ble
Model	Docu-mento	Docu-mento	Docu-mento Clave-valor	Columna	Columna	Columna	Clave-valor	Clave-valor	Grafos	Documento Clave-valor Grafo	Clave-valor	Colum-na
Integrity model	BASE	MVCC	ACID	ACID MVCC	BASE	MVCC	BASE	BASE	ACID	ACID BASE MVCC	ACID	MVCC
Atomicity	Condi-tional	Si	Si	Si	Si	Conditio-nal	Si	No	Si	Si	Si	Conditio-nal
ConSistency	Si	Si	Si	Si	Si	Si	Si	No	Si	Si	Si	Si
Isolation	No	Si	Si	No	No	Si	Si	Si	Si	Si	Si	Si
Durability	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si
Transactions	No	No	No	Si	No	Si	Si	No	Si	Si	Si	No
Referential integrity	No	No	No	No	No	No	No	No	Si	Si	No	No
ReviSion control	No	Si	Si	Si	No	Si	No	Si	No	Si	No	Si

Tabla 4 Características de indexado en sistemas NoSQL

DB	MongoDB	CouchDB	DynamoDB	HBase	Cassandra	Accumulo	Redis	Riak	Neo4j	OrientDB	Aerospike	Hypertable
Model	Documento	Documento	Documento Clave-valor	Columna	Columna	Columna	Clave-valor	Clave-valor	Grafos	Documento Clave-valor Grafo	Clave-valor	Columna
Secondary indexes	Si	Si	No	Si	Si	Si	No	Si	Si	Si	Si	Si
CompoSite keys	Si	Si	Si	Si	Si	Si	No	Si	No	Si	No	Si
Geospatial Indexes	Si	No	No	No	No	Si	No	No	Si	Si	No	No
Graph support	No	No	No	No	No	Si	No	Si	Si	Si	No	No

Tabla 5 Características de escalabilidad en sistemas NoSQL

DB	MongoDB	CouchDB	DynamoDB	HBase	Cassandra	Accumulo	Redis	Riak	Neo4j	OrientDB	Aerospike	Hypertable
Model	Documento	Documento	Documento Clave-valor	Columna	Columna	Columna	Clave-valor	Clave-valor	Grafos	Documento Clave-valor Grafo	Clave-valor	Columna
Horizontal scalable	Si	Si	Si	Si	Si	Si	Si	Si	No*	Si	Si	Si
Replication	Master-Slave Replica sets	Master-Slave (Continuous and one-shot)	Master-Slave	Master-Slave Master-Master Cyclic	Master-Master	Master-Slave Master-Master Cyclic	Master-Slave	Multi master	Master-Slave	Multi master	Synchronous	Master-Slave
Sharding	Si	Si	Si	Si	Si	Si	No	Si	Si	Si	Si	Si
Shared nothing Architecture	Si	Si	Si	Si	Si	Si	Si	Si	No	Si	Si	Si

Tabla 6 Lenguajes soportados en sistemas NoSQL

DB	Mongo-DB	CouchDB	DynamoDB	HBase	Cassandra	Accumulo	Redis	Riak	Neo4j	OrientDB	Aerospike	Hypertable
Language	Docu-mento	Docu-mento	Documento Clave-valor	Colum-na	Columna	Columna	Clave-valor	Clave-valor	Grafos	Documen-to Clave-valor Grafo	Clave-valor	Columna
C	Si	Si	No	Si	No	No	Si	No	No	Si	Si	No
C++	Si	Si	No	Si	Si	No	Si	No	No	Si	No	Si
C#	Si	Si	No	Si	Si	No	Si	Si	No	Si	Si	No
Clojure	No	No	No	No	Si	No	Si	No	No	Si	Si	No
Dart	No	No	No	No	No	No	Si	No	No	No	No	No
ColdFuSion	No	Si	Si	No	No	No	No	No	No	No	No	No
Erlang	Si	Si	Si	No	Si	No	Si	Si	No	No	Si	No
Go	Si	No	No	No	Si	No	Si	No	Si	No	Si	No
Groovy	No	No	Si	Si	No	No	No	No	Si	No	No	No
Haskell	Si	Si	No	No	Si	No	Si	No	No	No	No	No
Java	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si
JavaScript	Si	Si	Si	No	No	No	No	Si	Si	Si	No	No
Lisp	No	Si	No	No	No	No	Si	No	No	No	No	No
Lua	No	Si	No	No	No	No	Si	No	No	No	Si	No
.Net	No	No	Si	No	No	No	No	No	Si	Si	No	No
node.js	Si	No	No	No	Si	No	Si	No	No	Si	Si	No
Objective-C	No	Si	No	No	No	No	Si	No	No	No	No	No
OCaml	No	Si	No	No	No	No	No	No	No	No	No	No
Perl	Si	Si	Si	No	Si	No	Si	No	Si	No	Si	Si
PHP	Si	Si	Si	Si	Si	No	Si	Si	Si	Si	Si	Si
PL/SQL	No	Si	No	No	No	No	No	No	No	No	No	No
Python	Si	Si	Si	Si	Si	No	Si	Si	Si	Si	Si	Si
Ruby	Si	Si	Si	No	Si	No	Si	Si	Si	Si	Si	Si
Scala	Si	No	No	Si	Si	No	Si	No	Si	Si	No	No
Smalltalk	Si	Si	No	No	No	No	Si	No	No	No	No	No
TCL	No	No	No	No	No	No	Si	No	No	No	No	No

2.3 Sistemas de gestión de datos geográficos

Los datos geográficos se presentan como una información cuya naturaleza implica la creación de extensiones y/o programas específicos a la hora de desarrollar funcionalidades con componentes geográficas.

A pesar de tratarse como un campo más, el tipo de dato geográfico implica varias cuestiones más complejas que las que podría traer por ejemplo, un campo numérico. Es así por ejemplo con tipo de geometría, que puede ser de puntos, líneas o polígonos (en el caso del vectorial) y un sistema de referencia de coordenadas que es una cuestión crítica a la hora de trabajar con datos geográficos.

Sumado a lo anterior, las bases de datos geográficas suelen implicar una serie de funciones específicas, como las topológicas, que nos permiten trabajar de forma fluida con la información geográfica.

PostGIS

PostGIS es un módulo para el sistema gestor de base de datos relacional libre PostgreSQL, desarrollado Refractions Research Inc.

Este módulo proporciona a PostgreSQL la capacidad no solo de almacenar información geoespacial y cumplir la norma SFSS (Simple Features Specification for SQL), sino de dar la capacidad de realizar operaciones de análisis geográfico.

Además de la capacidad de manejar elementos geolocalizados, PostGIS ofrece muchas características que rara vez se encuentran en otras bases de datos espaciales de la competencia, tales como Oracle Locator / Territorio y SQL Server.

En la lista de capacidades de PostGIS podemos ver sus principales características [3]

Oracle Spatial/Locator

Con cada edición de Oracle Database se incluye Oracle Locator, que aporta las funciones y los datos cartográficos necesarios para habilitar por ubicación las aplicaciones empresariales.

Oracle Spatial [34], una opción de Oracle Database 11g, Enterprise Edition, es totalmente compatible con servicios web y 3D para gestionar toda la información geoespacial, incluidos datos vectoriales y ráster, topología y modelos de red. Está diseñada para satisfacer las necesidades de las aplicaciones de sistemas de información geográfica (GIS) avanzadas como gestión de terrenos, servicios públicos y defensa/seguridad interna.

El formato espacial nativo y abierto de Oracle en teoría elimina el coste de los sistemas específicos de propiedad exclusiva y es compatible con todos los principales productos GIS.

MySQL Spatial

MySQL Spatial [35] aparece basándose en la implementación del Open Geospatial Consortium (OGC) publica la norma OpenGIS de información geográfica.

Siguiendo la especificación OGC, MySQL implementa extensiones espaciales como un subconjunto del SQL con Geometría entorno de Tipos. Este término se refiere a un entorno SQL que se ha ampliado con un conjunto de tipos de geometría.

Una columna de SQL geometría de valor se implementa como una columna que tiene un tipo de geometría. La

especificación describe un conjunto de tipos de geometría SQL, así como las funciones en esos tipos para crear y analizar valores de geometría.

MySQL extensiones espaciales permiten la generación, almacenamiento y análisis de las características geográficas:

- Tipos de datos para representar los valores espaciales

- Funciones para la manipulación de valores espaciales

- Indexación espacial para la mejora de los tiempos de acceso a columnas espaciales

Microsoft SQL Spatial

En 2008 se celebra el lanzamiento de Microsoft de SQL Server que finalmente ofrece soporte Geoespacial [35] de la suite de productos de SQL Server.

Esto permite el almacenamiento de datos espaciales en tablas de SQL (en forma de puntos, líneas y polígonos) y un conjunto de funciones para permitir la manipulación de estos datos. También se incluyen nuevos índices espaciales para apoyar la ejecución de estas funciones.

SQL Server 2008 admite dos tipos de datos espaciales diferentes: la geometría y la geografía.

Este tipo de datos almacena los datos en superficies planas y proyectadas

GeoJinni

GeoJinni [37] (antes conocido como SpatialHadoop) es una extensión integral para Hadoop que permite el procesamiento eficiente de los datos espaciales.

Da la capacidad espacial en las diferentes capas y componentes de Hadoop para que sea más adecuado y más eficiente para almacenar y datos espaciales de errores proceso.

Más específicamente, se modifica la capa de almacenamiento, la capa de MapReduce y añade una nueva capa de operaciones.

En el nivel más bajo, que añade nuevos tipos de datos para los datos espaciales que pueden ser utilizados como claves o valores en un programa de MapReduce.

También agrega analizadores y escritores para la interacción con los archivos que contienen datos espaciales. A diferencia de Hadoop tradicional donde los datos en archivos están desorganizados, GeoJinni proporciona índices espaciales eficientes que se organizan en dos capas, índice global y el índice local.

Los datos de las particiones de índice globales a través de diferentes máquinas, mientras que los índices locales organiza los datos dentro de cada máquina. Este diseño se utiliza para construir tres índices diferentes en GeoJinni, grid tree, R y R + -Árbol. Todos estos índices se pueden construir a petición del usuario y se almacenan en el sistema de archivos distribuido Hadoop (HDFS).

Para permitir que los programas MapReduce para utilizar los índices construidos, añadimos dos nuevos componentes a la capa de MapReduce, a saber, SpatialInputFormat y SpatialRecordReader. El SpatialInputFormat utiliza el índice global por tabiques de archivos podas tempranas que son gama consulta externa. El desarrollador puede elegir qué particiones de carga basado en una función de filtro espacial que se define en el programa de MapReduce de la misma manera el mapa y reducir funciones se define. El SpatialRecordReader utiliza el índice local permite la función de mapa para elegir sólo un subconjunto de registros a procesar en lugar de procesar todos los registros

en un bloque.

Los nuevos componentes en la capa de MapReduce permite GeoJinni para ejecutar muchas operaciones espaciales de manera eficiente mediante la utilización de los índices subyacentes. La capa de operaciones encapsula todas las operaciones espaciales para permitir a los usuarios finales acceder a ellos de forma rápida y eficiente.

GeoMesa

Geomesa [38] aparece como parte de un esfuerzo para migrar una analítica predictiva espacial a un entorno de cloud computing, se descubrió que el apoyo a la infraestructura espacial de almacenamiento, consulta, y la transformación que habíamos llegado a depender de PostGIS y Geoserver fue en gran parte inexistente en la nube.

En GeoMesa se implementó lo suficiente de la base espacial necesaria para apoyar la tarea inmediata de la migración de la analítica predictiva a un entorno operativo Hadoop / Apache Accumulo.

Al mismo tiempo, se dedicó a la implementación de un esquema de indexación espacio-temporal para incorporar estos conjuntos de datos en nuestro enfoque de modelado.

2.4 Librerías para la visualización en 3D

Los globos virtuales han tenido un impacto positivo en muchas empresas tecnológicas de España y el resto del mundo, tanto en la organización económica, política y en relación con la imagen corporativa.

También ha tenido un impacto positivo en las industrias relacionadas con la gestión de datos geográficos. Se han percibido cambios en las estrategias de negocio en la industria nacional sobre las indicaciones geográficas como resultado de los globos virtuales. Además, la tecnología de las aplicaciones geográficas también ha tenido un impacto positivo en la educación.

Dentro del terreno del software de visualización 3D en formato de software libre podemos encontrar muchas aplicaciones disponibles que implementan diferentes algoritmos de nivel de detalle (Level of Detail o LOD), gestión del terreno y otros aspectos de este tipo de navegadores. Tal y como ya hemos comentado, el objetivo de estas aplicaciones ha sido siempre la navegación en 3D por terrenos geográficos virtuales. Si bien sus comienzos estuvieron orientados a los juegos de ordenador, muchos de ellos encontraron una gran utilidad en la navegación virtual por zonas geográficas concretas. Entre estas aplicaciones podríamos destacar las siguientes:

- Cesium [39] Cesium es una librería JavaScript para crear globos en 3D y mapas 2D en un navegador web sin necesidad de un plugin. Utiliza WebGL para gráficos acelerados por hardware, y es multiplataforma, cross-browser, y en sintonía para visualización de datos dinámico. Cesium es de código abierto bajo la licencia Apache 2.0. Es gratuito para uso comercial y no comercial.

- OssimPlanet [40] es un módulo desarrollado en C++, construido sobre el software Ossim, que permite la visualización geoespacial realista en 3D, a partir de un modelo digital de terreno en formato SRTM, y la inclusión de capas remotas a través del protocolo WMS.

- WorldWind [41] es un globo virtual desarrollado por la agencia espacial NASA, originalmente escrito en Java, que ofrece fotografía satélite y aérea y mapas topográficos, así como inclusión de elementos 3D accesibles desde servidores remotos.

- Virtual Terrain Project (VTP) [42] es un proyecto software que pretende promover la creación de herramientas para la construcción de cualquier parte del mundo real en forma digital, interactiva y tridimensional, relacionando tecnologías de áreas diversas como diseño asistidos por ordenador (CAD), sistemas de información geográficos (SIG), simulación visual y teledetección.

- Glob3Mobile (G3m) [48] es u proyecto multiplataforma (web, iOS y android) que trabaja con webGL para la visualización de los datos en 3D.

2.5 Aplicaciones de gestión portuaria

A continuación se citan una serie de aplicaciones existentes de gestión portuaria.

Debido a lo específico de las funcionalidades es de SmartPort, no es comparable íntegramente con estas aplicaciones, ni ellas entre sí, pero si no todas, presentan algunas tareas en común.

Galeón

Galeón es un Sistema para la gestión de la Señalización Marítima de un ente portuario. Su objetivo principal es el control de la señalización, tanto de señales en tierra como flotantes, dentro del área portuaria.

Permite las siguientes funcionalidades:

Inventariar las señales marítimas, tanto a nivel textual como geoespacialmente.

Definición de los parámetros de intervención de las señales en función de su tipología.

Registro de intervenciones realizadas sobre las señales y de averías producidas.

Histórico de intervenciones y situación de señales.

Informes de análisis sobre intervenciones, señales y avisos sobre el mapa, ratios.

Carga directa de cartografía utilizada por departamento de delineación.

Funcionamiento sobre explorador web sin necesidad de instalaciones en cliente.

Posibilidad de integración con sistema de comunicación de balizas: avisos de avería, posicionamiento en mapa a partir de coordenadas suministradas, etc.

gisport/gisweb

GISPORT y GISWEB son dos familias de productos dirigidos a la gestión gráfica de Autoridades Portuarias.

GISPORT consiste en un conjunto de aplicaciones Windows orientadas a la utilización por parte de usuarios internos de departamentos como Explotación, Seguridad, etc.

GISWEB, sistema que utiliza tecnología Web para acceder a través de un navegador estándar a información gráfica relacionada con la explotación.

Dentro de la familia GISPORT se distinguen los siguientes módulos con sus principales características:

GISPORT-Atraques:

Gestión gráfica de autorizaciones, registro de atraques y desatraques, situación gráfica de buques, mapas temáticos, previsiones de atraque, histórico de buques.

GISPORT-Concesiones:

Representación gráfica de bienes y expedientes de concesión y autorización.

GISPORT-Ocupaciones:

Representación gráfica de las ocupaciones de mercancía en muelle.

GISPORT-SAFE (GISPEM):

Gestión de planes de emergencia del puerto.

e-stiba

Aplicación de inerza para la gestión del entorno portuario y la gestión.

Plantea una aplicación de software para la gestión GIS de atraques, gestión del dominio público, centro de intercambio de mensajes, etc... son desarrollos que en su momento INERZA a emprendido logrando la modernización y automatización de los procesos portuarios.

Marinawin

Marinawin es una aplicación que busca optimizar las tareas íntegras de la gestión portuaria.

Entre las diferentes funcionalidades que incorpora el programa se encuentran:

- Gestión de estancias

- Gestión de varaderos

- gestión de locales

- Módulo para la facturación

Infoport

Nuestras soluciones para Autoridades Portuarias incluyen desde sistemas de información para la gestión a sistemas de automatización de procesos o la integración de sistemas de información y control.

- Gestión de mercancías y pasaje

- Valoración y facturación de servicios

- Operaciones portuarias

- Gestión del dominio publico

- Seguridad y control

- Mercancías peligrosas

- Conservación

- Talleres

- Comercial

Paquetes de Portel

Productos GIS desarrollados por PORTEL y GTD , orientados a facilitar la gestión gráfica de distintos aspectos de una gestión portuaria.

La aplicación GIS diseñada para la publicación y consulta gráfica de atraques a través de Internet, dirigida a Autoridades y Comunidad Portuaria.

El sistema cuenta con los siguientes módulos:

1. GisEsc para la gestión de escalas y atraques

2. GisCon para la gestión de los distintos expedientes concesionales existentes en un puerto así como sus bienes

3. GisMMPP para la gestión de las mercancías peligrosas ubicadas en el ámbito operacional del organismo gestor del puerto

4. GisSec orientado a ser una herramienta para gestionar los distintos aspectos de la seguridad del organismo gestor del puerto

3. Descripción de SmartPort

Un puerto se presenta como una frontera dual entre las comunicaciones del medio marino y terrestre, así como una fusión del mundo antrópico y natural.

Dentro del entorno natural, uno de los objetivos principales de un puerto es controlar el estado del medio, para ello, es necesario disponer de una red de sensores conectados a Internet o Internet de las cosas, también conocido como Internet of Things (IoT).

Disponer de una red de sensores de mayor o menor tamaño nos permite monitorizar y reaccionar frente a situaciones adversas, tales como la aproximación de una serie de olas con una altura elevadas o la existencia de fuertes rachas de viento, por lo que tener un buen refresco de datos es un elemento fundamental en cualquier aplicación de este tipo.

Además, la zona de influencia del Puerto de la Luz de las Palmas de G.C. interactúa con elementos de la ciudad de Las Palmas de G.C., como sucede con los emisarios de aguas residuales. Por lo que tener un seguimiento y un pronóstico marino/costero puede ser de interés para los departamentos de alcantarillado y municipales de protección del medio ambiente [43].

Así mismo, es importante obtener información fiable y frecuente sobre uno de los principales elementos de un puerto: los barcos. Es crítico controlar la entrada y salida de los mismos, así como su tipo, carga, procedencia y, especialmente, su localización geográfica.

Todos estos sensores, insertando nuevos registros en periodos muy cortos de tiempo generan grandes volúmenes de información (Big Data) [44], cuyo principal objetivo, no es sólo el almacenaje de la información, además busca cruzar la información para obtener posibles modelos predictivos que permitan la previsión de fenómenos no deseables en la zona portuaria.

Fenómenos que se desean controlar como la posibilidad de extensión de un posible vertido, que mediante los corrientímetros se podría averiguar la evolución posible de una mancha. Situación muy últil por ejemplo lo que sucedió en 15 de Marzo de 2015 con el pesquero ruso Opleg Naydenov que fue arrastrado fuera del puerto de Las Palmas al no poderse apagar el incendio que sufría a bordo. Este pesquero se ha hundió a 20 millas (24 kilómetros) al sur de Maspalomas (Gran Canaria).

Mediante herramientas de monitorización y sensorización adecuada, se pretende poder actuar y analizar frente a situaciones como estas.

Figura 2. Mancha producida por el vertido del pesquero Oleg. Fuente: la Provincia

Los elementos susceptibles de ser analizados en un puerto, por todo lo comentado, pueden ser variables como:

Medio antrópico:

- Infraestructuras portuarias; construcciones terrestres y marítimas.
- Posiciones del barco, tipo de barcos, rutas y horarios.
- Muelles disponibles.
- Mercancías y pasajeros transportados.
- Lugares de interés cercanos: Transportes, hospitales, otros servicios ...
- Posibles fugas de petróleo.
- Activos de gestión de emergencias.

Medio natural:

- Orografía y modelos digitales del terreno.
- Batimetría.
- Niveles de altura de agua y mareas.
- Movimientos de las olas. Dirección, frecuencia y altura.
- Salinidad del agua.
- Análisis químico del agua
- Biosfera marina. Presencia de peces, cetáceos, algas....
- Espacios naturales protegidos

De todas estas necesidades nace SmartPort; una aplicación web enriquecida que aporta un sistema de visualización y gestión de los datos del Puerto de la Luz de Las Palmas de Gran Canaria.

Una de los hitos principales de SmartPort es mostrar los datos que generan los sensores disponibles en el entorno del puerto, así como su histórico [45].

Además, el sistema mostrará la ubicación geográfica de todos los recursos disponibles para la autoridad portuaria así como otros actores que están en el puerto y en sus alrededores.

El proyecto SmartPort nace mediante un acuerdo de colaboración entre la Universidad de las Palmas de Gran Canaria (ULPGC) y la Autoridad Portuaria de las Palmas, todo bajo el paraguas del proyecto que es la plataforma Internet del futuro de la arquitectura de la Comunidad Europea (FIWARE) [46], en el que la ULPGC es socio en diversos paquetes de trabajo del proyecto.

FIWARE viene a determinar el primer pilar tecnológico del proyecto. Es una plataforma que permite la validación de nuevos conceptos, tecnologías, modelos de negocio, aplicaciones y servicios a gran escala. Otra de las principales características de FIWARE es la existencia de módulos o Generic enablers (GE) [47], que dan respuesta a cada necesidad específica de forma atomizada y modular.

El segundo pilar tecnológico de SmartPort es el globo virtual Glob3mobile [48], desarrollado por la compañía de software IGO y la Universidad de Las Palmas de Gran Canaria [49]. Esta librería nos proporciona las funcionalidades de visualización en 3D.

En el presente trabajo se explica la arquitectura y los componentes individuales de la aplicación SmartPort.

Sin duda uno de los elementos más importantes de una aplicación son los datos; las interfaces van y vienen en el tiempo y las tendencias de diseño son muy volátiles en su recorrido. Pero un modelado y estructura de datos permanecen cuasi inalterables en el tiempo. Por lo que organizar, estructurar y aportar escalabilidad a los datos es crítico a la hora de definir una aplicación informática.

En SmartPort tenemos varias fuentes de información, las cuales las hemos diferenciado en datos dinámicos y estáticos. (Véase figura 3).

Los datos estáticos son aquellos que tienen una actualización baja en el tiempo ya que depende, por ejemplo, de la apertura de nuevos hoteles. En el caso del puerto tenemos datos estáticos que se han digitalizado desde diversas fuentes de datos como Google Earth o referencia proveniente de las IDEs (Infraestructuras de datos espaciales), tales como:

- Infraestructuras

- Hoteles

- Restaurantes

- Cajeros automáticos

- Paradas de guaguas

- Farmacias

Por el contrario los datos dinámicos son aquellos que tienen un refresco muy alto como pueden ser los datos que provienen de las boyas y de los sensores meteorológicos con un refresco cada 3 minutos, y que son, a su vez, una de las fuentes principales de generar grandes volúmenes de datos.

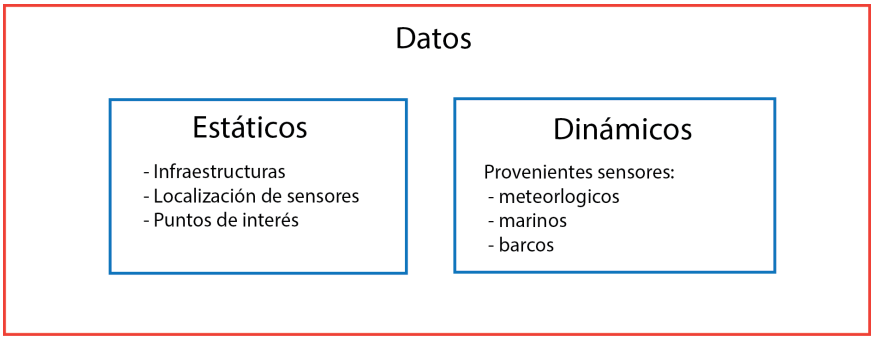


Figura 3. Fuentes de datos en SmarPort

Cuando hablamos de la naturaleza de los datos dinámicos, podemos determinar varios dispositivos hardware.

1) Sensores meteorológicos.

– Geonica 41001

– Geonica 05106

– Geonica 52203

Figura 4. Imagen del sensor Geonica 52203

2) Sensores corrientímetros.

– Aandela Instruments series 3791-3798

– Geonica Datamar 2000C

Figura 5. Imagen del sensor Aandela Instruments series 3791-3798

3) Barcos

– Provenientes del sistema AIS (Automatic Identification System.

Figura 6. Imagen del AIS Simrad AI150

4. BackEnd. Descubiendo la tecnología base del proyecto

4.1 FIWARE

La base de SmartPort está en la tecnología proporcionada por FIWARE , una infraestructura innovadora abierta basada en la nube para la creación y entrega de futuras aplicaciones y servicios de Internet rentable, a gran escala.

Presenta una serie de API (Application Programming Interface) accesible de forma pública y libre, impulsado por el desarrollo de una implementación de referencia de código abierto que acelera la disponibilidad de productos y servicios comerciales basados en tecnologías FIWARE.

FIWARE contiene una serie de elementos modulares o Generic Enablers que son fácilmente internconectables entre si y que los podemos diferenciar en las siguientes familias:

1.-Datos de contexto

2.- IoT

3.-Procesado en tiempo real

4.- APIs de control de acceso

5.- Publicación de datos. Open Data

6.- Análisis de Big Data

7.- Creación de dashboards

8.- Procesado de streams en tiempo real

9.- Experiencia de usuario avanzada

10.- Alojamiento de aplicaciones en la nube

4.2 Datos estáticos

Los datos estáticos, debido a su relativa sencillez, no requieren de una arquitectura compleja; sólo necesitan un medio de almacenaje tradicional. Es el caso de los sistemas gestores de bases de datos relacionales (SGBDR) y de un operario que digitalice la información tal y como aparece en la figura 7.

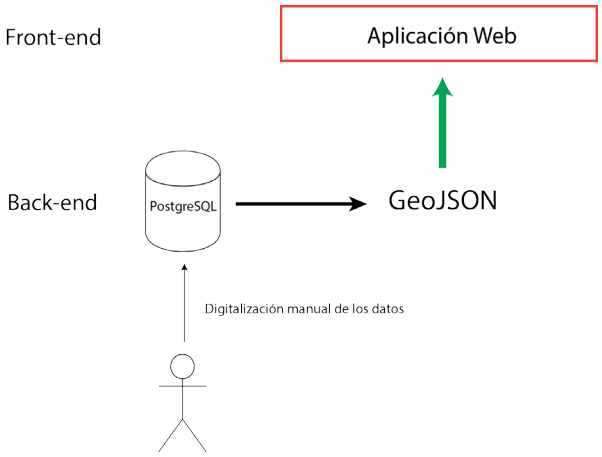


Figura 7. Arquitectura y secuencia de trabajo para los datos estáticos

La secuencia de trabajo a la hora de trabajar con datos estáticos es la siguiente:

1.- Los datos son digitalizados por un operario mediante un sistema de información geográfica (SIG), en este caso hemos usado QGIS.

2.- El propio SIG edita y almacena los datos en un SGBDR. El SGBDR elegido es PostgreSQL con su extensión para manejo de datos geográficos PostGIS.

3.- Convertimos y almacenamos los datos que están almacenados en la base de datos a GeoJSON para poder visualizarlos en el visor. Formato para la codificación de una variedad de estructuras de datos geográficos que soporta los siguientes tipos de geometría: Point, LineString, polígono, multipunto, MultiLineString y MultiPolygon. Las listas de geometrías están representados por un GeometryCollection.

En la figura 8 vemos un ejemplo de formato de GeoJSON

```
{
        "type": "Feature",
        "geometry": {
         "type": "Point",
        "coordinates": [125.6, 10.1]
 },
 "properties": {
        "name": "Dinagat Islands"
            }
}
```

Figura 8. Formato GeoJSON

4.3 Datos Dinámicos

Dentro de los datos dinámicos encontramos aquellos que vienen de un servicio de FTP en el cual, todas las mañanas a las 7.00 am actualiza los datos de las entradas y salidas de barcos en el puerto.

Ese archivo lo parseamos mediante un script de Python, para almacenarlo en una base de datos y convertirlo a JSON y lo mostramos en la aplicación.

Además, en esa base de datos tenemos una galería en construcción con las fotos de los barcos que disponemos, de este modo se pueden mostrar en la parte frontal de la aplicación.

Para los barcos de los cuales no tenemos fotos, ofrecemos una serie de imágenes genéricas según el tipo de barco que sea (carguero, pasajero, petrolero). Información que es suministrada por los ficheros txt que se encuentran en el FTP.

Un poco más complejo, es el tratamiento de datos provenientes de los distintos sensores en el Puerto de la Luz es uno de los elementos sobre el cuales gira la aplicación que hemos desarrollado.

Actualmente capturamos los siguientes datos:

1) Procedente de las boyas:

Tabla 7. Datos procedentes de las boyas

1	Altura significativa espectral
2	Altura de ola significante
3	Altura media del 10% de olas máximas
4	Altura de la ola máxima
5	Período medio de todas las olas
6	Periodo de Pico del oleaje
7	Periodo medio de paso por el cero ascendente
8	Dirección media en el pico=dirección de olas de mayor energía
9	Dispersión de la dirección del oleaje en el pico de energía
10	Dirección media de procedencia del oleaje
11	Índice de unidireccionalidad
12	Presión de la columna de agua sobre el sensor AWAC
13	Velocidad orbital en superficie sobre el sensor AWAC
14	Dirección de la corriente en superficie
15	Periodo de pico (olas dominantes) del oleaje.
16	Velocidad de la corriente
17	Temperatura del agua

18	Salinidad del agua

2) Procedente de los sensores meteorológicos

Tabla 8: Datos procedentes de los sensores meorológicos

1	Temperatura
2	Presión
3	Humedad relativa
4	Lluvia
5	Visibilidad media
6	Visibilidad mínima
7	Radiación solar

3)Procedente del AIS

Tabla 9 Datos procedentes del AIS

1	Fecha
2	Nombre
3	Eslora
4	Nacionalidad
5	Tipo de Operación
6	Muelle de atraque
7	Compañía
8	Fecha de llegada
9	Puerto de Origen
10	Código de puerto
11	Tipo de buque
12	Crucero tránsito
13	Código Lloyd
14	Fecha de salida

15	Bolardos del muelle de atraque
16	Puerto de Destino

Una de las principales características de estos datos es la frecuencia de refresco, estando establecido en una media de cada 3 minutos. Esto implica un rápido crecimiento de la información almacenada en la base de datos, por lo que hemos determinado que un sistema tradicional de base de datos no da respuesta a nuestras necesidades.

Aquí es donde entra en juego FIWARE, gracias a su arquitectura modular con los Generic Enablers, ha resultado muy sencillo la implementación de nuevas arquitecturas, siendo una de sus grandes virtudes la eliminación de tiempo dedicado a instalación y configuración del sistema.

Si recordamos los objetivos de la aplicación, tenemos dos grandes elementos que deberían estar separados en la arquitectura: La obtención del último valor del sensor, y de los históricos.

Para obtener los últimos valores que nos da cada sensor, lo hacemos mediante el GE Orion [50]. Orion tiene un sistema de subscripciones al sensor que nos permite almacenar y consultar los últimos datos disponibles. Se basa en MongoDB; un sistema de bases de datos tipo clave-valor que almacena la información en formato JSON [51].

Orion nos provee de interfaces que aportan la posibilidad de interactuar mediante dos API REST [52]: NGSI9 [53] y NGSI10 [54].

Gracias a ellas, somos capaces de realizar operaciones tales como:

NGSI 9:

- Registrar contenido

- Descubrir la disponibilidad del contexto

- Subscribirse a la disponibilidad del contexto

- Actualizar disponibilidad del contexto

- Desinscribirse de la disponibilidad del contexto

NGSI 10:

- Actualizar contenido

- Consultar contenido

- Subscribirse al contenido

- Actualizar subscripción

- Desinscribirse del contenido

Una vez obtenido esto, para una interactuación ágil con la aplicación web, y para simplificar las transacciones HTTP, se generó una capa de abstracción encima de Orion, y, de este modo poder realizar consultas a los datos de forma más amigable mediante peticiones AJAX [55]. (Véase figura 9).

Gracias a esto, mediante una consulta sencilla podemos obtener los datos de los sensores. Un ejemplo para obtener los datos de los barcos sería:

http://<<IP>>:<<Port>>/orion/query.html sensorType=ship&sensorID=.&pattern=true

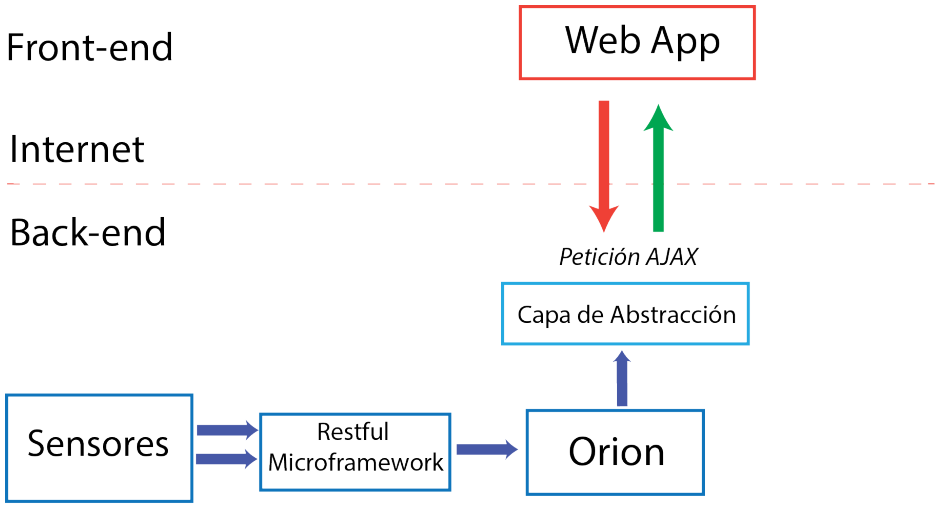


Figura 9: Arquitectura de datos dinámicos

Como se planteó en los objetivos, uno de los hitos es obtener los datos históricos de los sensores, para ello usamos el GE Cosmos [56] Cosmos es una implementación para el trabajo con *Big Data*, permitiendo el despliegue de clústers computacionales privados basados en el ecosistema de Hadoop.

Cosmos permite a los usuarios:

• Operaciones de E/S. Un sistema de clústers de persistencia basados en HDFS

• Creación, uso, y borrado de clústers, basados en MapReduce y consultas similares a SQL mediante Hive o Pig .

• Gestionar la plataforma, en muchos casos como servicios, usuarios, clusters, etc, desde la API y la línea de comandos de Cosmos.

En nuestra aplicación, como se muestra en la figura 7, usamos los datos existentes en Orion para darles persistencia y almacenarlos en Cosmos, para ello usamos Cygnus [57], otro GE basado en Flume [58], que es un canal de flujo de datos distribuido, fiable, y que proporciona un servicio disponible para la recoger, agregar, y mover grandes cantidades de datos de manera eficiente.

Cygnus tiene una arquitectura simple y flexible basada en el streaming de flujo de datos. Se utiliza un modelo de datos extensible simple que permite una aplicación analítica que se encarga de procesar esa información y almacenarla en HDFS, mediante el uso de la API WebHDFS.

Todos estos GEs se encuentran localizados en la infraestructura física de FIWARE, que recibe las peticiones desde un servidor de la ULPGC, donde está ubicada la aplicación web.

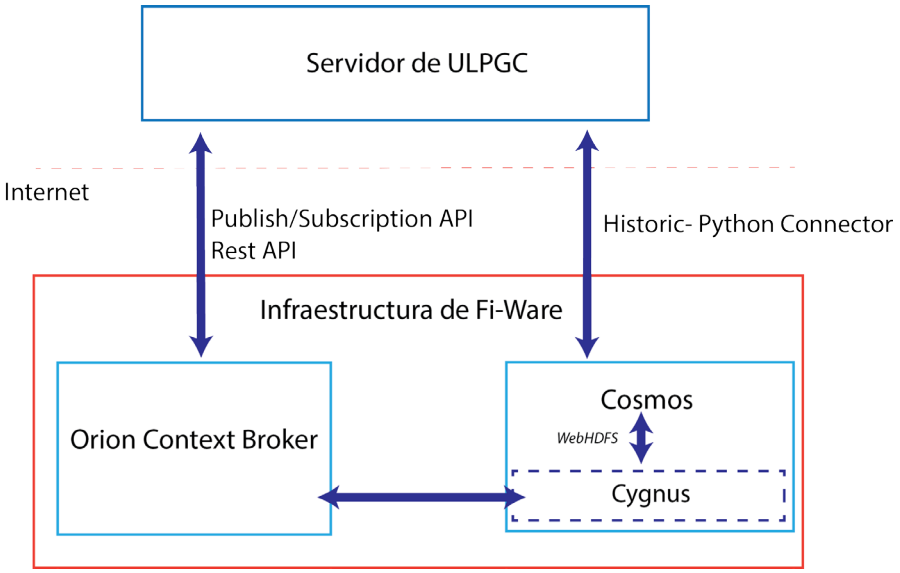


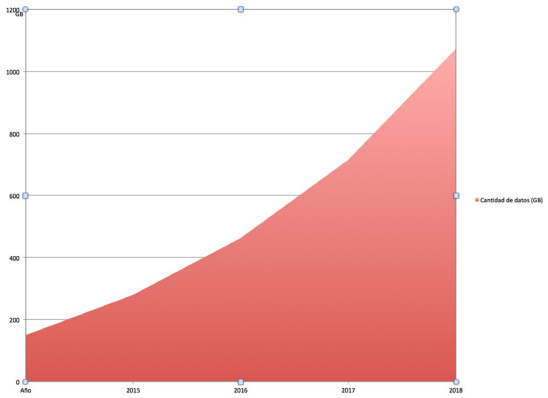
Figura 10: Arquitectura general de trabajo con *BigData*

Una de las decisiones de adoptar los módulos de Big Data es la escalabilidad en el tiempo. Actualmente se generan una cantidad de datos de aproximadamente 140 Mb/día, lo que implicaría, un aumento exponencial de datos a lo largo del tiempo, incluyendo la previsión de incorporar nuevos sensores.

Por lo tanto, para la correcta previsión de futuro es indispensable contar con un ecosistema de gestión de grandes volúmenes de datos. En la tabla 10 y en la gráfica 1 se muestra la previsión de crecimiento en 5 años.

Tabla 10: Previsión de cantidad de datos en 5 años

Año	Cantidad de datos (GB)
2015	150
2016	280
2017	462
2018	716
2020	1072



Gráfica 1: Previsión de cantidad de datos en 5 años. Datos provenientes de la tabla 10.

Tener, relativamente pocos datos actualmente, produce una serie de problemas a la hora de operar con la información.

Uno de los handicaps de Hadoop es la baja velocidad de consulta, de hecho, al estar preparado para grandes volúmenes de datos, no ofrece sus capacidades más óptimas a la hora de trabajar con pequeñas cantidades de datos.

Por lo que en comparación a un sistema gestor de bases de datos relacional, la petición de datos es lenta, como figura en la tabla que aparece a continuación en la tabla 11:

Tabla 11: Tiempo de consulta en diferentes SGBD.

Tiempo (ms)	MySQL	PostgreSQL	Hadoop
Todos los datos meteorológicos	3,03	4,99	59,51
Un atributo ordenado por fecha	1,55	2,53	74,83

Para solucionar este problema hemos procedido a generar una arquitectura específica de gestión de datos, en la cual, diferenciamos en tres capas diferentes:

• Capa Batch: Donde se almacenan todos los datos mediante Hadoop. Esta capa precalcula las vistas conforme los datos se puedan ir procesando. El postprocesado de todos los datos van a la capa de servicio.

• Capa de Servicio: El rol principal de esta capa es dar respuesta a las consultas de manera eficiente; mediante un SGBDR se almacenan los resultados de las consultas y son almacenadas en esta capa.

• Capa de velocidad: Uno de los mayores inconvenientes que hay que solucionar es el procesado del streaming de los datos. Gracias a esta capa, los datos más recientes pueden ser incluidos en los resultados de los análisis mientras se propagan en el sistema distribuido de Hadoop.

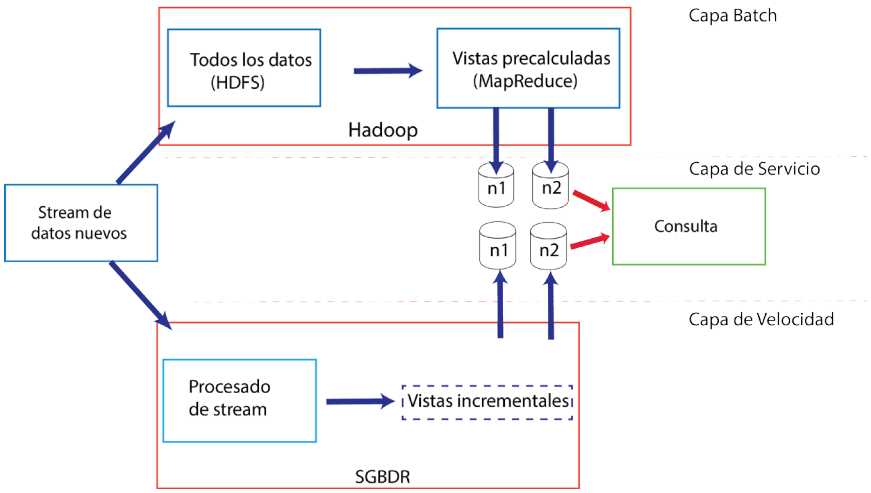


Figura 11. Arquitectura propuesta para la explotación de grandes volúmenes de datos

Una vez implementada la arquitectura los resultados obtenidos son los que muestran en la tabla 12:

Tabla12 : Velocidad de consulta en diferentes SGBD con arquitectura en capas

Tiempo (ms)	MySQL	PostgreSQL	Hadoop optimizado
Todos los datos meteorológicos	3,03	4,99	4,76
Un atributo ordenado por fecha	1,55	2,53	5,98

4.4 Sistema de alertas

Las alertas se presentan como un elemento estratégico en SmartPort. Actualmente palabras como BigData y Smart se usan a la ligera siguiendo las corrientes de modas y, según nuestro punto de vista, para que una aplicación sea Smart, debe permitir un mínimo de analítica para poder tomar decisiones y es donde aparecen las alertas.

El sistema de alertas permite avisar al usuario cuando un sensor tome un valor determinado, y de este modo controlar el sistema de sensores del puerto.

El sistema de alertas comienza en la creación de una alerta. Esta alerta se determina por el tipo de sensor sobre el que se va a crear la alerta, el atributo que se quiere tener en cuenta, si queremos que salte cuando sea igual, mayor o menor que el valor que determinemos.

En el backend creamos una subscripción mediante una API propia desarrollada en Python, para que vaya consultando en Orion mediante la interfaz NGSI si se llega a un valor determinado.

Esta alerta está almacenada en una base de datos MongoDB.

Por otro lado, cuando una alerta salta, se almacena en una base de datos MySQL y, mediante una API propia en PHP se muestra en la aplicación web. Se mostrarán tanto las alertas que están en este momento activas, como

las que han saltado en un momento dado pero no se están cumpliendo.

En la figura 12 se muestra el esquema de la arquitectura

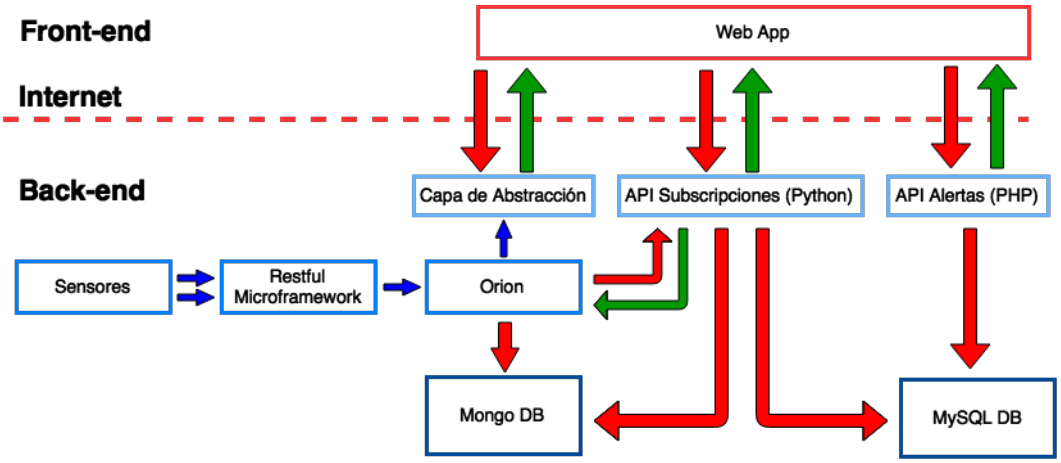


Figura 12 Arquitectura del sistema de alertas

Algunos ejemplos de código de estas alertas son los siguientes:

- API de Subscripciones (subsApi.py)

API escrita en Python que se encarga de la gestión de las subscripciones de las alertas. Nos ayuda a no repetir suscripciones y abstrae a las aplicaciones clientes en cierta medida del formato de las mismas. Contiene, entre otras, las siguientes funciones:

-checkAndCreateSubs: Función que comprueba si existe una subscripción para un atributo de un sensor determinado:

-Si existe: procede a actualizarla si es preciso.

- Si no existe: crea una suscripción nueva

```
def checkAndCreateSubs(sensorType, sensorID, attribute):
  client = MongoClient(conf.orion['mongoDB']['host'], conf.orion['mongoDB']['port'])
  db = client['orion']
  exist= db['csubs'].find_one({'reference': conf.consumer['url'],
                               'entities':  {'$elemMatch':
                                 { 'type' : sensorType,
```

```
                                    'id': sensorID
                                    }
                                },
                                'attrs': attribute
    })
  if (None != exist):
    return updateSubs( str( exist["_id"] ), attribute)
  return createSubs(sensorType, sensorID, attribute)
```

createSubs: Función que crea una nueva suscripción a un atributo de un sensor específico. Encapsula el formato y la configuración para realizar una suscripción.

```
def createSubs(sensorType, sensorID, attribute):
  json_element= {
    "entities":[
      {
        "type": sensorType,
        "isPattern": "false",
        "id": sensorID
      }
    ],
    "attributes":[ attribute ],
    "reference": conf.consumer["url"],
    "duration": conf.consumer['duration'],
    "notifyConditions" : [
    {
      "type": "ONCHANGE",
      "condValues": [
        attribute
      ]
```

```
        }
        ], “throttling”: “PT5S”
    }
  json_data =json.dumps(json_element)
  # convert str to bytes
  post_data = json_data.encode(‘utf8’)
  # we should also say the JSON content type header
  headers = {}
  headers[‘ContentType’] = ‘application/json’
  headers[‘Accept’] = ‘application/json’
  # now do the request for a url
  req = urllib2.Request(conf.orion[‘subs’][‘urlService’], post_data, headers)
  # send the request
  res = urllib2.urlopen(req)
  response=res.read()
  res.close()
  return response
```

dbConnections.php

Funciones auxiliares en PHP que permiten fácil manejo de las alertas almacenadas en la base de datos MySQL.

addAlert: Función que añade una alerta. Si ya existe, devuelve un json con un error indicándolo. En caso contrario devuelve el identificador de la nueva alerta insertada.

```
function addAlert($sensorType, $sensorId, $attribute, $condition, $value, $alias)
{
   $mysqli= connection();
   if ($mysqli>connect_errno) {
      die (json_encode(array(“error”=> “Cannot connect to MySQL: “ . $mysqli
>connect_error))) ;
   }
   $stmt =  $mysqli>prepare(“SELECT id FROM alerts where  (deleted=0 and sensorType = ? and sensorId = ? and
attribute = ? and  cond = ?  and value = ? and alias <=> ? )”);
```

```
    $stmt >bind_param('ssssds', $sensorType, $sensorId, $attribute, $condition, $value, $alias);
    if(! $stmt>execute())
    {
        die (json_encode(array("error"=> "Cannot get Alerts: " . $mysqli>error))) ;
    }
    $stmt >store_result();
    if($stmt>num_rows > 0)
    {
        $stmt>close();
        die (json_encode(array("error"=> "Alert already exists"))) ;
    }
    $stmt>close();
    $stmt =  $mysqli>prepare("INSERT INTO alerts(sensorType, sensorId, attribute, cond,
value, alias) VALUES (?, ?, ?, ?, ?, ?)");
    $stmt >bind_param('ssssds', $sensorType, $sensorId, $attribute, $condition, $value, $alias);
    if(! $stmt>execute())
    {
        die (json_encode(array("error"=> "Cannot insert Alert: " . $mysqli>error))) ;
    }
    $id=$mysqli>insert_id;
   $stmt>close();
   $mysqli>close();
   return $id;
}
```

removeAlert: Función que borra una alerta con el identificador especificado.

```
function removeAlert($id)
{
    $mysqli= connection();
    if ($mysqli>connect_errno) {
        die (json_encode(array("error"=> "Cannot connect to MySQL: " . $mysqli
```

```
>connect_error))) ;
   }
   $stmt =  $mysqli>prepare("DELETE FROM alerts WHERE id = ?");
   $stmt >bind_param('i', $id);
   if(! $stmt>execute())
   {
      die (json_encode(array("error"=> "Cannot delete Alert: " . $mysqli>error)))
   }
  $stmt>close();
  $mysqli>close();
  return true;
}
```

iFrameCommunicationAPI.js

API JavaScript que permite la comunicación de elementos situados en distintos iframes en SmartPort.

registerInput: Función que permite registrar un punto de entrada para recibir mensajes. Y procesarlos con la una función especificada. Comprueba que las entradas estén habilitadas así como que el punto de entrada no exista actualmente. Crea una variable para almacenar el mensaje así como otra que sirve para saber si un elemento ha sido modificado.

```
function registerInput(inputName, action){
   if($("#inputsZone").length == 0)
{
      //it exists!
      throw "Inputs aren't enabled. You must enabled it before use registerInput";
      return 1;
   }
   if($("#" + inputName).length != 0) {
      //it exists!
      throw "Input already exists! You must delete it before registrate it again";
```

```
      return 1;
   }
   var newInput= $("<input type ='' id='"+inputName+"' name='iframeInput' value='' >");
   try
   {
    $('#inputsZone').append(newInput);
   }
 catch (err)
   {
    throw "Cannot create input";
    return 2;
   }
   var newInput2= $("<input type ='hidden' id='"+inputName+"Changed' name='iframeInput' value='false' >");
   try
   {
    $('#inputsZone').append(newInput2);
   }
   catch (err)
   {
    throw "Cannot create input";
    return 2;
   }
   if( bind(inputName, action))
   {
    return 3;
   }
    return 0;
}
```

bind Permite vincular una acción cuando se reciba un mensaje en la entrada especificada. Borra las vinculaciones anteriores.

```
function bind(inputName, action){
```

```
try
{
 $("#"+inputName).unbind();
 $("#"+inputName).on('inputEvent',function(){ action($("#"+inputName).val());});
      console.log($("#"+inputName));
}
catch (err)
{
 throw "Cannot bind action";
 return 1; }}
```

messageBetweenIframes Función que permite enviar un mensaje al punto de entrada deseado del iFrame indicado. Acepta JSON o strings.

```
function messageBetweenIframes(targetIframe, inputName, message)
{
 message=messagePreProcess(message);
 try
 {
   $('#'+targetIframe,
window.parent.document).contents().find("#"+inputName).val(message);
 parent.document.getElementById(targetIframe).contentWindow.$('#'+inputName).trigger('inputEvent');
    return 0;
 }
 catch (err)
 {
  console.log("Cannot send message: "+ err);
  return 1;
 } }
```

5. Frontend: la tecnología más cercana al usuario

5.1 Primera versión: Wirecloud

En la primera versión elegimos Plataforma Wirecloud [59] como entorno de desarrollo en *frontend*, ya que nos permitía desarrollar la aplicación como componentes independientes y para posteriormente interconectarlo, y, además de estar en en ecosistema FIWARE, nos aportaría una interconexión óptima a la hora de integrar los diferentes GEs de.

GEs utilizados y Arquitectura

• Orion Contexto Broker

• Plataforma Mashup Wirecloud

• Cosmos Big Data

• Cloud Platform Fiware

Infraestructura y equipamiento

Con el fin de implementar nuestros servicios, utilizamos en esta fase del proyecto las infraestructuras suministradas por Plataforma Wirecloud y FIWARE como se ve en el diagrama en la sección anterior.

En cuanto a sensores, que no tienen acceso directo a ellos, por lo que necesitan para leer los valores de una base de datos proporcionada por la Autoridad Portuaria de Las Palmas. En ese momento sólo teníamos acceso a la información sobre medidores de corriente y estaciones meteorológicas.

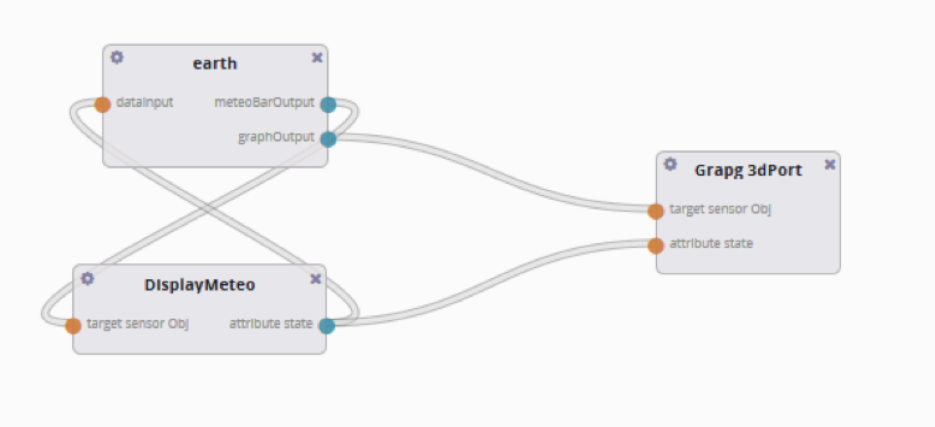


Figura 13: interconexión de widgets en Wirecloud

Pruebas y Validación

El *front-end* de la aplicación fue probada usando navegadores Chrome y Firefox instalados en Ubuntu de escritorio, Windows 8 y OSX 10.5. Desde algunas de las características WebGL no son compatibles ni en Internet Explorer (IE) ni navegador web Safari, nuestra aplicación no ofrece soporte para ellos.

Para probar los servicios de *back-end*, también desarrollamos una serie de scripts escritos en Python que simulan todo el proceso de datos.

Durante el desarrollo de la aplicación encontramos las siguientes dificultades:

- NGSI API cambió cuando Wirecloud se actualiza, por lo que algunas correcciones se tuvieron que hacer para que el operativo de la aplicación.

- El proceso de depuración duro: a la hora de tratar depurar los widgets creados funcionalidad es necesario crear una nueva versión del widget cada vez que queríamos probar un error, y resultaba era un poco molesto e improductivo.

- A veces Wirecloud no reconocía la nueva versión de un widget y, en lugar de la nueva, la versión anterior se demostró, por lo que hemos perdido tiempo tratando de adivinar si tuvieras la última de ellas en el área de trabajo.

- Las API basadas en Ajax tiene muchas ventajas, pero también tienen desventajas en tiempo de depuración: los puntos de interrupción no funcionaban dentro de un onSuccess () función / onFail () dentro de la petición Ajax, así que tuvimos que depurar su contenido por la forma antigua , usando impresiones.

- La Plataforma Mashup Wirecloud no permite la creación dinámica de widgets.

- Falta de documentación avanzada sobre Plataforma Wirecloud.

- No existe una API para hacer consultas a Cosmos por Plataforma Wirecloud.

- Con el fin de acceder a los datos almacenados en el Cosmos de máquinas situadas fuera de la red de confianza, hemos tenido que desarrollar un servicio intermedio asignado en una máquina virtual FIWARE Cloud.

- Tuvimos problemas con Cygnus almacenar datos dentro del cosmos como documentos JSON. El problema era que nuestros servicios de Python que acceden remotamente Cosmos solamente fueron capaces de leer el formato CSV. Para superar este problema, hemos modificado el código fuente Cygnus para adaptarse a nuestro caso y guardar en formato CSV con una columna adicional para permitir la identificación de los registros como un bloque.

Por lo tanto, decidimos prescindir de Wirecloud para futuras versiones de SmartPort.

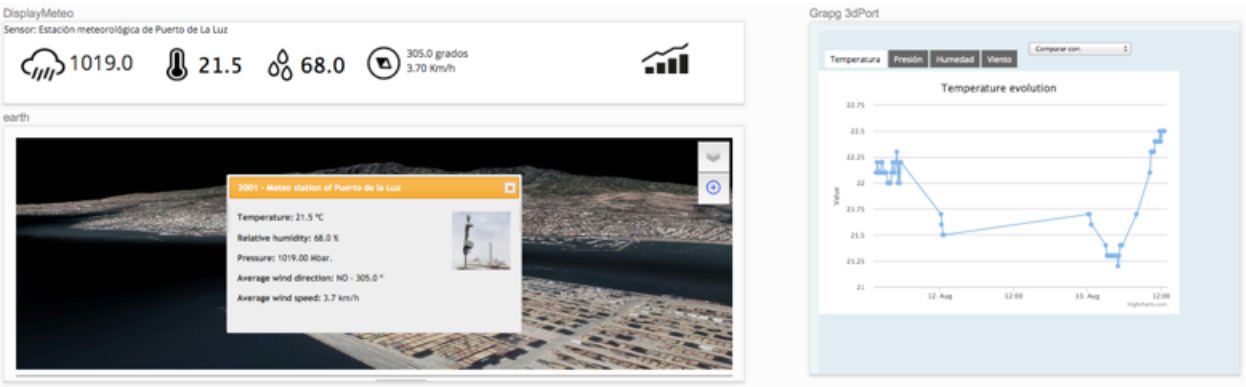


Figura 14: Primera versión de SmartPort con Wirecloud

5.2 Segunda versión: HTML 5 + Javascript

La versión basada en web con G3m el cual lo hemos incrustado como un widget en el *front-end*.

Este elemento puede ser generado a partir de un proyecto de aplicación web, basado en GWT y escrito en Java. Esta aplicación web permite dos formas principales de interacción con el desarrollador y el resto de los elementos de la interfaz.

La API G3m y códigos fuente están disponibles en Java. Por lo tanto, el mundo virtual es totalmente configurable a partir de los parámetros de inicialización del código de aplicación web, el establecimiento de cosas como la imagen deseada, los controles de navegación o la posición inicial de la cámara.

Es posible también para desarrollar toda la funcionalidad necesaria para nuestro visor en Java y para que sea accesible al navegador mediante consolidaciones de Javascript. Esto permite comportamientos dinámicos del globo virtual controlados por otros elementos presentes en la interfaz de usuario.

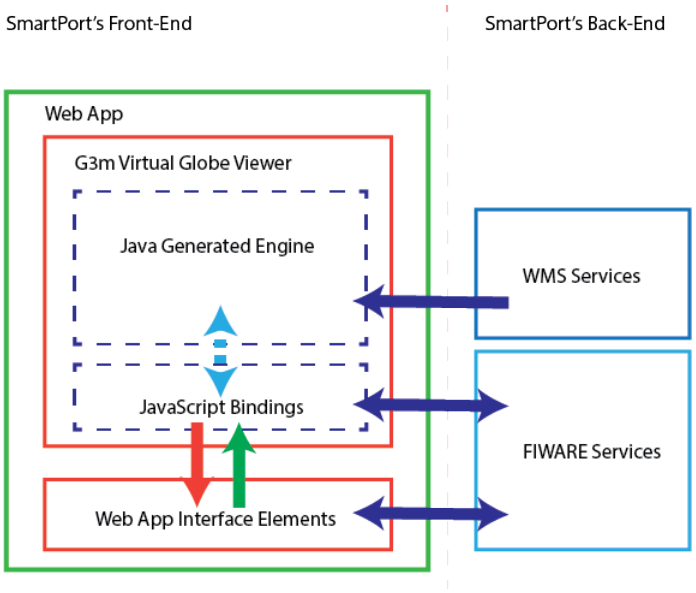


Figura 15 Arquitectura de integración entre G3m y SmartPort

Cabe destacar que la principal diferencia de G3m dentro del código abierto mapas 3D de la comunidad es su enfoque multiplataforma.

En este momento, la metodología de implementación nos permite añadir nuevas capacidades para la programación del motor principal en C ++. Estos cambios se aplicarán directamente a todas las plataformas, lo que significa iOS, Android y Web.

El proceso de implementación se basa en las siguientes herramientas:

- Kit de desarrollo de iOS

- Convertidor C ++ a Java

- Android SDK

- Google Web Toolkit (GWT)

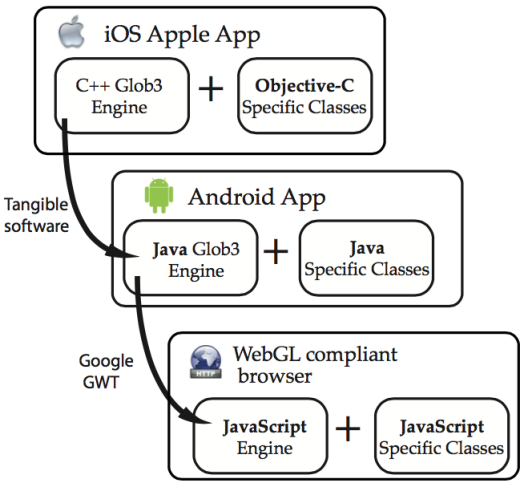


Figura 16 Proceso de implementación de G3m

A pesar de la interfaz de usuario actual de SmartPort está basada en Web, un punto clave para el futuro es el soporte de plataformas móviles como Android y iOS.

Aunque enfocar el proyecto en plataformas móviles implica tener en cuenta una serie de elementos como un soporte nativo para la interacción multitáctil.

Por lo tanto, y buscando tener una escalabilidad del proyecto la interacción multitáctil está integrado en nuestra web front-end y se puede utilizar desde dispositivos con pantallas capacitivas.

5.2.1 Visualización del terreno en 3D

Desde un punto de vista funcional, el motor globo virtual debe proporcionar ciertas características para apoyar las necesidades de funcionamiento de la interfaz de SmartPort.

Una característica clave es mostrar diferentes capas de información del terreno en función del interés particular del usuario.

Hay muchas maneras de añadir imágenes georeferenciadas. En nuestro caso, hemos decidido utilizar el protocolo Web Map Service (WMS) desarrollada por la comunidad Open Geospatial Consortium. El uso de este protocolo basado en HTTP, en el cual podemos obtener imágenes para cubrir nuestra representación de la superficie del globo.

Estas imágenes pueden ser procesadas, tales como callejeros, mapas de delimitaciones administrativas, visualización LiDAR o fotografías aeroespaciales como imágenes de satélite.

En nuestro caso, la aplicación SmartPort utiliza la fuente de las imágenes del proyecto Virtual Earth. Este conjunto de imágenes es proporcionada por Microsoft y cubre todo el planeta con alta resolución de fotografías aéreas.

Sin embargo, una aplicación destinada para representar los elementos presentes en el entorno portuario requiere una aún mayor resolución de la imagen. Es por ello, los alrededores del puerto de la Gran Canaria se usan imágenes se captan de un servicio local WMS. En este caso, SmartPort utiliza el servicio WMS de GRAFCAN, una empresa pública responsable del mantenimiento de los datos geográficos de las Islas Canarias.

Estas dos capas formarán el conjunto de imágenes de base predeterminada que define el aspecto del terreno.

Además de esto, el sistema G3m permite combinar estas imágenes de base con otras capas semitransparentes para enriquecer el contenido de información.

5.2.2 Carga de servicios interoperables

Otros servicios WMS también puede proporcionar estas imágenes añadidas. En este caso, el proyecto SmartPort permite al usuario combinar información de los mapas batimétricos. Estos mapas hacen fácil para el usuario final para tomar decisiones bien informadas en relación con rutas o calado máximo de los buques.

Además, el sistema proporciona también por defecto una capa de los espacios naturales. Estas imágenes pueden ser superpuestas sobre la imagen base, mostrando importantes espacios naturales en los alrededores del puerto.

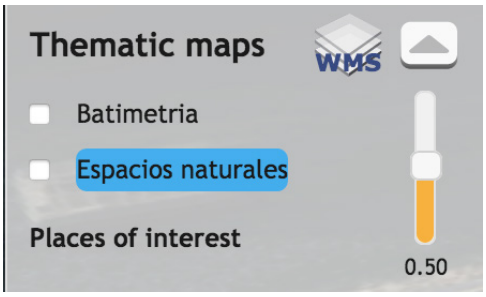


Figura 17 WMS disponibles por defecto y selección de transparencia

Por otro lado, y buscando dar flexibilidad a la aplicación, se ha agregado la posibilidad de añadir servicios WMS en dinámico, es decir, que el usuario pueda poner sus propios servicios interoperables WMS.

Esto se consigue realizando una petición GetCapabilities al servidor estándar y mediante la respuesta que nos da, en formato XML, seremos capaces de conocer qué capas nos suministra el servicio.

Un ejemplo podría ser hacer la petición al servicio de encuesta de infraestructuras y equipamientos locales de Canarias.

Grafcan nos suminsitra una URL de acceso al servicio:

http://idecan2.grafcan.es/ServicioWMS/EIEL

Y mediante esa URL, debemos mandar dos comandos para poder saber qué capas están disponibles:

- Servicio, en este caso WMS

- Request, en este caso GetCapabilities

Por lo tanto, la petición que haríamos al servidor quedaría algo así:

http://idecan2.grafcan.es/ServicioWMS/EIEL?service=WMS&request=getcapabilities

Esto nos devuelve un XML que parseamos y procesamos para conocer las capas que hay disponibles

```xml
▼<WMS_Capabilities xmlns="http://www.opengis.net/wms" xmlns:sld="http://www.opengis.net/sld" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:ms="http://mapserver.gis.umn.edu/mapserver" version="1.3.0" xsi:schemaLocation="http://www.opengis.net/wms http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd
 http://www.opengis.net/sld http://schemas.opengis.net/sld/1.1.0/sld_capabilities.xsd http://mapserver.gis.umn.edu/mapserver http://idecan2.grafcan.es/ServicioWMS/EIEL?
 service=WMS&version=1.3.0&request=GetSchemaExtension">
  <!--  6.0.0 (vg 1.7) GRAFCAN 2011 -->
 ▶<Service>...</Service>
 ▼<Capability>
  ▶<Request>...</Request>
  ▶<Exception>...</Exception>
   <sld:UserDefinedSymbolization SupportSLD="1" UserLayer="0" UserStyle="1" RemoteWFS="0" InlineFeature="0" RemoteWCS="0"/>
  ▼<Layer>
     <Name>WMS_EIEL</Name>
    ▶<Title>...</Title>
     <Abstract>EIEL Propiedad del Gobierno de Canarias.</Abstract>
    ▶<KeywordList>...</KeywordList>
     <CRS>EPSG:32628</CRS>
     <CRS>EPSG:4326</CRS>
     <CRS>EPSG:32627</CRS>
    ▶<EX_GeographicBoundingBox>...</EX_GeographicBoundingBox>
     <BoundingBox CRS="EPSG:32628" minx="164622" miny="2.94455e+06" maxx="685418" maxy="3.37115e+06"/>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
    ▶<Layer queryable="1" opaque="0" cascaded="0">...</Layer>
   </Layer>
  </Capability>
</WMS_Capabilities>
```


Figura 18 Resultado de un GetCapabilities.

Tomamos las capas disponibles para que el usuario sea capaz de seleccionar las que quiere añadir al visor, y una vez seleccionada, empezamos a hacer la petición de teselas georreferenciadas. Los comandos que se utilizan en este caso son:

- Capa deseada
- Bounding Box
- Coordenadas

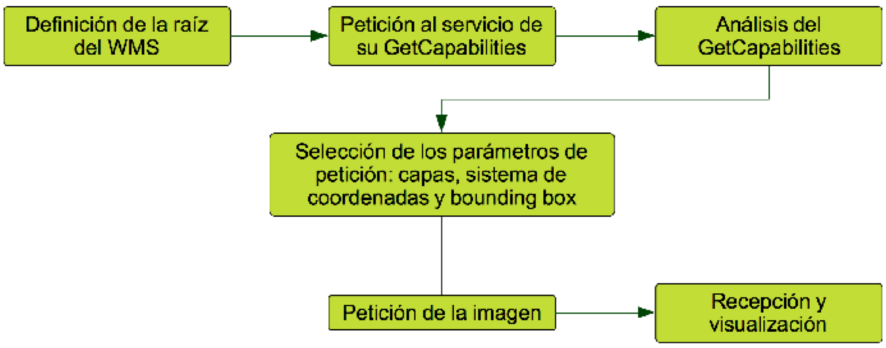


Figura 19 Flujo de petición de WMS

Una vez se tengan esos parámetros, el globo se encarga de hacer la petición para poder poner las teselas en el propio visor.

El flujo de trabajo en la aplicación se muestra a continuación:

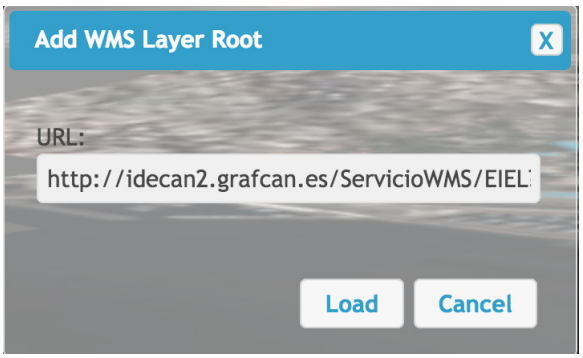


Figura 20 Selección del WMS

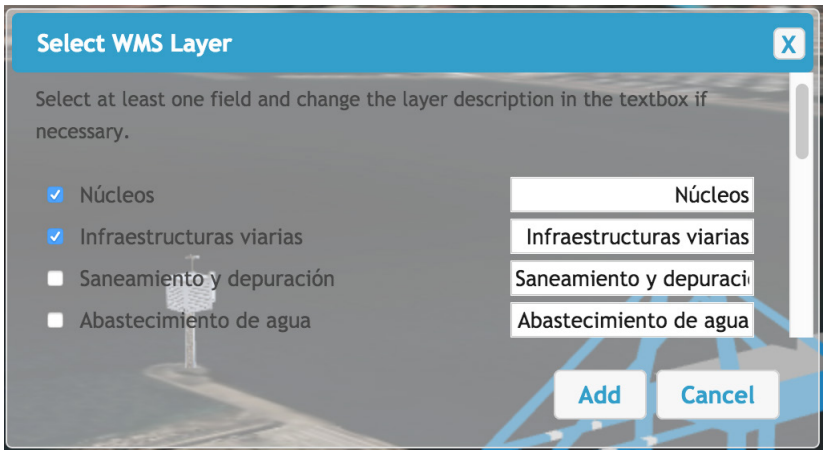


Figura 21 Selección capas disponibles para cargar en el globo

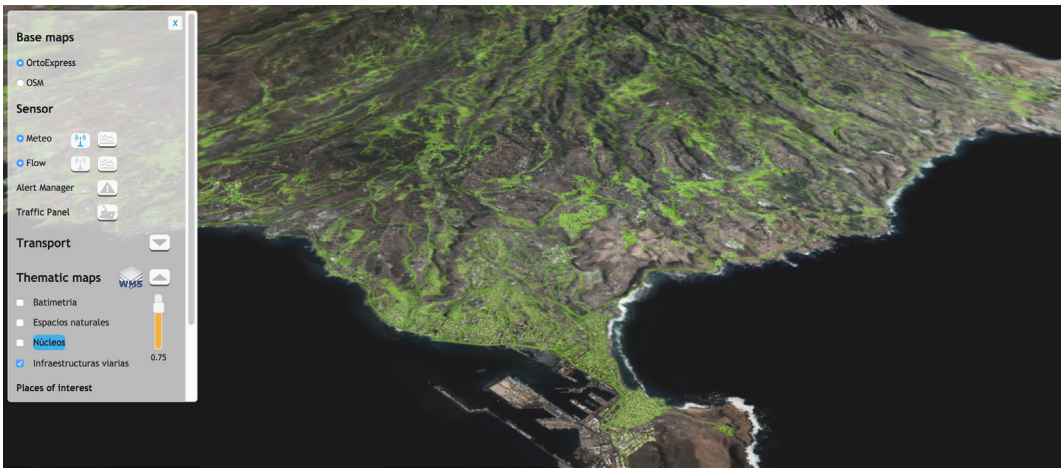


Figura 22 Visualización de capas WMS

5.2.3 Modelo de elevaciones

Un modelo de elevación puede parecer poco relevante en un sistema de control de puerto. Sin embargo, los puertos son a menudo rodeados por la orografía que hace fácil para sus usuarios a localizar a sí mismos y los elementos cercanos.

Este efecto es notable en un paisaje montañoso como Gran Canaria. Además, la inclusión de un modelo de elevación hace que al usuario le resulte más fácil de usar la aplicación.

En el caso de Gran Canaria, se ha utilizado un elevaciones mapa regular de valores 1e6 para dar forma a la isla. Este mapa se carga durante la inicialización de la interfaz de usuario como un único archivo en Band Interleaved por Line (BIL).

Esto permite que el sistema sea independiente de cualquier servidor de datos de elevación en línea, lo que hace que los cambios en el terreno más rápido para llevar a cabo.

El sistema G3m permite el uso de múltiples fuentes de datos de elevación en diferentes resoluciones.

Esto permite al desarrollador utilizar elevaciones muy detallados archivos sólo en zonas de interés y el uso de aproximaciones más amplias en otras áreas. Por otra parte, las zonas de fricción, tales como el mar se pueden visualizar sin necesidad de cargar los datos de elevación.

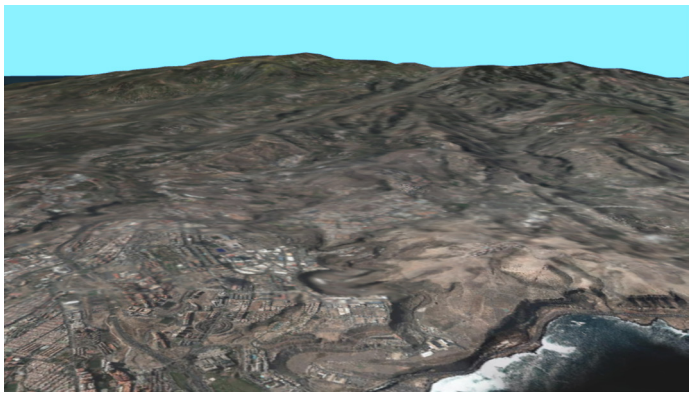


Figura 23 Visualización de capa del terreno

5.2.6 Navegación del globo

Como se ha indicado anteriormente, el motor de control de la cámara G3m permite multitouch así como la navegación del ratón / teclado.

Estos mecanismos de control son suficientes para colocar la cámara en cualquier lugar adecuado que el usuario lo desea, puede visualizar. Esta posición de la cámara está parametrizada por su ubicación geográfica (latitud y longitud), su altura sobre el terreno y el encabezamiento y el tono de la dirección de la vista sobre el terreno.

Sin embargo, el sistema de control del teclado del ratón podría ser desconocido para el usuario. En este caso, G3m nos permite mostrar widgets interactivos 2D en la parte superior de la vista 3D que permiten al usuario controlar la cámara. El front-end SmartPort ha añadido dos controladores en pantalla que permiten al usuario gestionar la altura de la cámara, su título y el tono de la vista.

El segundo es un widget arrastre bola que permite mover la dirección de la cámara en la misma dirección que se arrastra el centro del widget. La dirección arrastrando tiene dos componentes: X, lo que altera el rumbo de la partida a la cámara, e Y, que afecta el tono de la vista actual. Este control permite explorar la escena desde una localización estática.

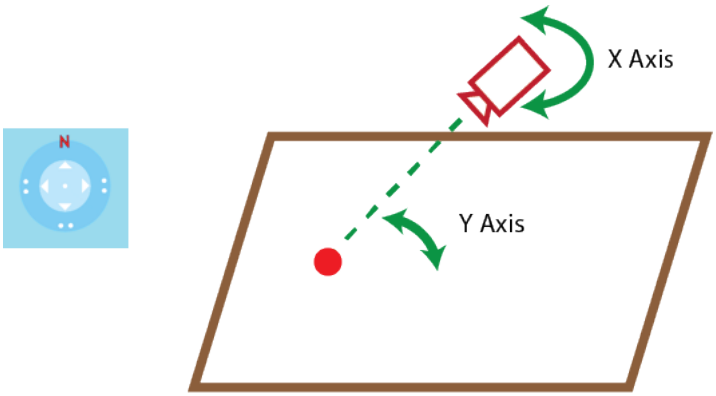


Figura 24 Inclinación de la cámara

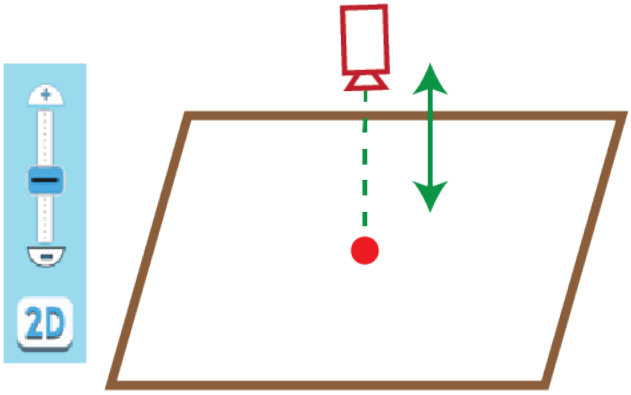


Figura 25 Aumento/disminución de Zoom y cambio 2D/3D

5.2.4 Marcas 3D

A pesar de la naturaleza 3D del entorno Glob3M, a veces la forma más clara para mostrar la ubicación de un elemento dentro de la escena es simplemente usar un icono 2D, o marca, que se encuentra en una posición fija.

La razón para preferir una marca 2D puede ser que el elemento para representar es demasiado complejo o que se encuentra en un sólo lugar.

Una de las funcionalidades de la SmartPort es mostrar los puntos de interés que están cerca del puerto. Estos puntos de interés son simplemente lugares que si el usuario lo desea, puede buscar en el mapa, así que no hay representación física para ellos. Además, en un uso típico podría ser de hasta 350 marcas en la escena, que deben elaborarse rápidamente.

El *pipeline* de representación deG3m permite dibujo miles de marcas en una velocidad de fotogramas adecuada.

Se utiliza una técnica *billboarding* basado en la pipe programable de móviles en GPU que evita la necesidad de mantener un modelo 3D para cada marca.

Además, el SDK G3m añade una capa de abstracción que impide al usuario hacer frente a la programación de la GPU (Trujillo, 2014). De esta manera, la prestación de numerosas marcas no aumenta considerablemente la velocidad de dubujado.

Figura 26 Renderizado de múltiples marcas.

Con respecto a la generación de iconos para las marcas, el proyecto G3m proporciona una API multiplataforma de canvas para la creación de estas imágenes.

Esta API nos permite generar "sobre la marcha" iconos que combinan diferentes imágenes o incluso añadir texto a las marcas. De esta manera el sistema es capaz de determinar si el usuario está en contacto con una marca determinada.

Cada vez que se hace clic se realiza la acción correspondiente. En este caso, el sistema muestra un cuadro de diálogo informativo web, que describe la naturaleza de la entidad representada por la marca.

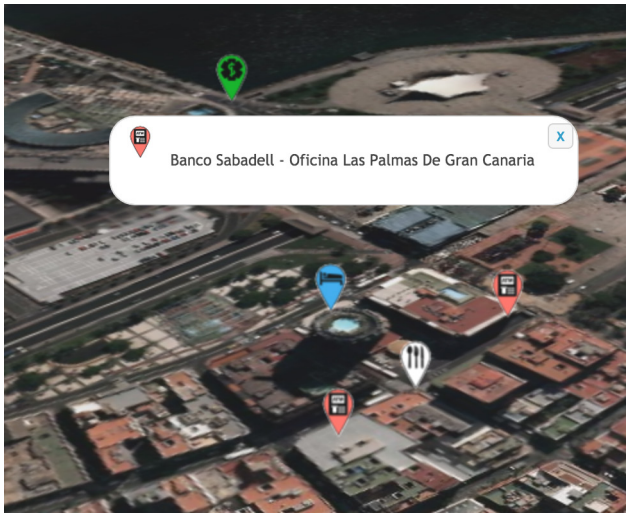


Figura 27 Interacción con marca

5.2.5 Modelos 3D

SmartPort también utiliza modelos 3D para representar los activos importantes dentro de la zona portuaria.

Estas entidades son las boyas, los sensores marítimos, los barcos, las grúas y los contenedores. Los usuarios pueden fácilmente reconocer estos elementos por su posición dar forma.

En el ejemplo de Gran Canaria hemos localizado cinco boyas en las proximidades del puerto. Cada una de ellos está representada por su propio modelo 3D.

El motor G3m permite al desarrollador utilizar modelos texturizados y renderizarlos utilizando la luz y de sombra efectos.

Como ocurre con las marcas, es de gran interés que todos los objetos en 3D de la escena se pueden seleccionar para que podamos realizar acciones y mostrar información como el usuario interactúa con ellos. En el caso de SmartPort para web, esta interacción podría ser mediante el uso de la dispositivo del ratón o las pantallas multitáctiles.

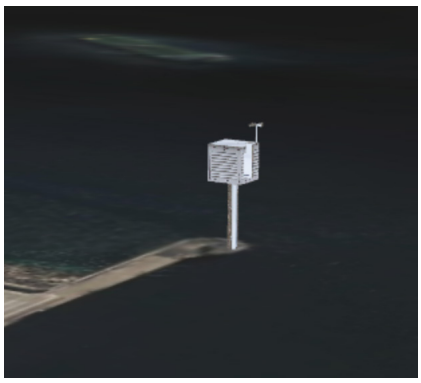


Figura 28 Modelo 3D de un sensor meteorológico

Figura 29 Modelo 3D de una boya

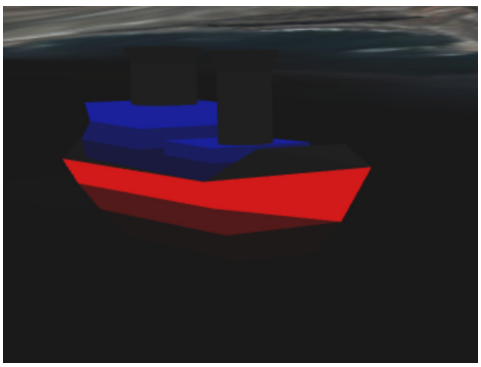


Figura 30 Modelo 3D de un barco de tipo genérico

Posteriormente, en SmartPort creímos que una funcionalidad muy interesante dentro del entorno portuario es la interacción con las grúas y contenedores. Debido a que es una de las fuentes de ingresos económico más importantes, unido al transporte de pasajeros, ubicamos modelos en tres dimensiones para mostrar una funcionalidad futura que podría tener la aplicación.

Figura 31 Modelo 3D de una grúa de contenedores

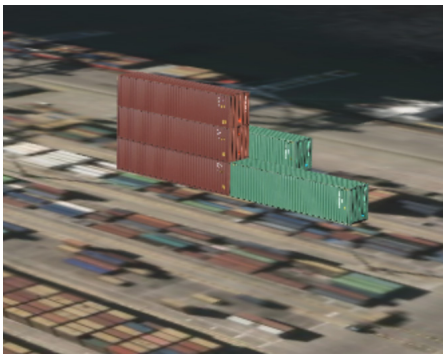


Figura 29 Modelo 3D de contenedores

La aplicación de este criterio de selección es sobre la base de la caja de delimitación jerarquías que permiten informáticos de manera eficiente el problema intersección. Una vez que el objeto ha sido seleccionado, se envía un mensaje al resto del sistema. De esta manera una determinada acción se pudo realizar, como se pudo mostrar datos recogidos por el sensor seleccionado

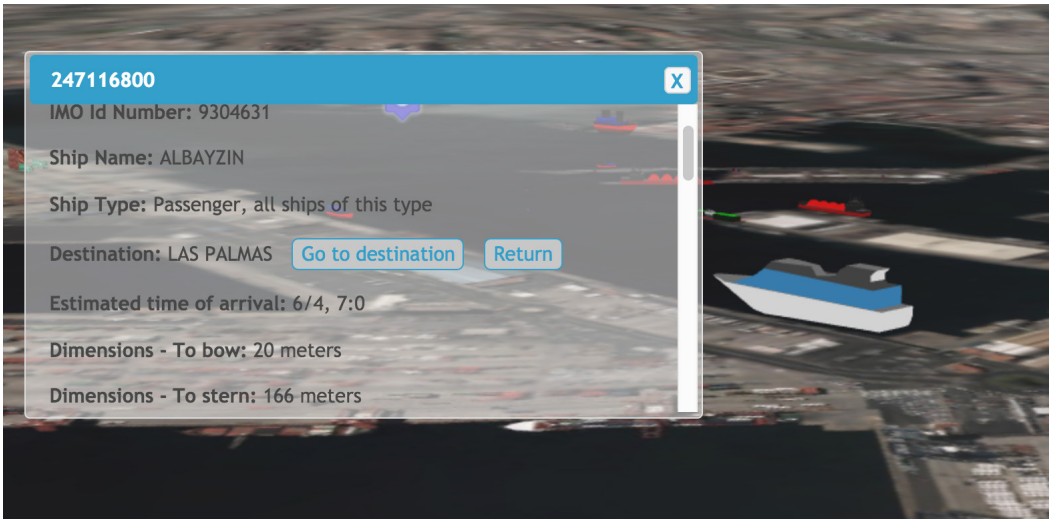


Figura 33 Interacción con un modelo de un barco

5.2.7 Muestra de datos estáticos

En la tabla de contenidos disponemos de un acceso a los datos estáticos que presenta SmartPort. Estos datos, debido a su naturaleza, sólo se actualizan cada vez que se realiza una modificación en la base de datos, siendo la carga de ducha información estática cada vez que se carga la aplicación en formato GeoJSON.

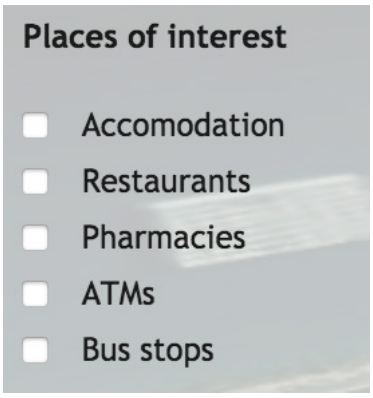


Figura 34 Carga de lugares de interés en la TOC

5.2.8 Muestra de datos dinámicos

Mediante la arqutectura definida en el apartado Back End, los datos dinámicos son mostrados gracias a la API realizada en Python en SmartPort.

Cada vez que se obtienen nuevos datos, estos cambian, ya que constantemente se están mostrando en la parte superior de la aplicación.

Debido a que tenemos diferentes naturalezas sensores y con diferentes localizaciones geográficas, en la interfaz hemos optado a realizar la siguiente implementación:

- En la tabla de contenidos podemos decidir de qué sensor tomamos los datos en vivo.

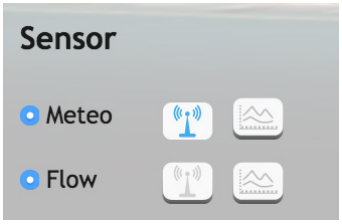


Figura 35 Selección de fuente de datos de sensor

-Según donde se encuentre el usuario navegando dentro del globo, en la parte superior de la aplicación mostrará los datos del sensor más cercano.

Figura 36 Muestra de los datos del sensor más cercano

5.2.9 Panel de E/S de barcos

Dentro de los datos dinámicos citamos a los datos de las entradas y salidas diarias de los barcos.

Para acceder a dicha información disponemos de un botón en la TOC que nos permite mostrar la ventana resumen de todas los movimientos de ese día en el puerto.

Para ello ofrecemos a modo resumen la lista de las entradas y salidas, además de una caja de texto por si se desean hacer búsquedas.

Figura 37 Panel de e/S de barcos

Si el usuario hace clic en una línea de la lista aparecerá información extendida del barco correspondiente.

Figura 38 Panel de E/S extendido

5.2.10 Datos de barcos en vivo

Mediante el AIS y gracias a un flujo de streaming que proporciona el Puerto de Las Palmas, somos capaces de conocer el estado actual de diversos barcos del puerto.

El acceso al AIS lo tenemos vía HTTP, donde tenemos un flujo de datos con un patrón determinado que somos capaces de parsear. Entre otras variables, en el AIS capturamos:

- IMO (International Maritime Organization) id

-Origen

- Destino

- Tipo de barco

- Dimensiones

- Última localización conocida

- Estimated time of arrival: 12/20, 8:0

- AIS Version: ITU 1371

- DTE: Estado del Data terminal

- Positioning information

- Tipo de mensaje que emite

- Última emisión

```
        });
}

function queryForBClassAISPosition(){
    $.ajax({ url:orionIP+'ais_position_class_b'+"&sensorID="+'.*'+"&pattern=true"+"&_"+new Date().getTime(),
        dataType: 'json',
        accepts: {
            json: 'application/json'
        },
        headers: {
            "Content-type": 'application/json',
            "Accept": 'application/json'
        }})
        .done(function(data) {
            parsingAISResponse(data,"ais_position_class_b");
        })
        .fail(function(data) {
            //alert("Request error: "+JSON.stringify(data));
        });
}

function queryForBClassStatic(){
    $.ajax({ url:orionIP+'ais_static_class_b'+"&sensorID="+'.*'+"&pattern=true"+"&_"+new Date().getTime(),
        dataType: 'json',
        accepts: {
            json: 'application/json'
        },
        headers: {
            "Content-type": 'application/json',
            "Accept": 'application/json'
        }})
        .done(function(data) {
            parsingAISResponse(data,"ais_static_class_b");
            showAISObjects();
            window.glob3m_aisArrayData(AClassAISPositionObjects,0);
            window.glob3m_aisArrayData(BClassAISPositionObjects,1);
            window.glob3m_aisArrayData(AClassAISStaticObjects,2);
            window.glob3m_aisArrayData(BClassAISStaticObjects,3);
        })
        .fail(function(data) {
            //alert("Request error: "+JSON.stringify(data));
        });
}
```

Figura 39 Sample de código del parseador del AIS

Gracias a esta información, mediante la aplicación de SmartPort, somos capaces de renderizar modelos en 3D según el tipo de barco, y sus dimensiones.

A su vez, localizamos y orientamos el barco según la información que nos suministre el AIS y tenemos un sistema de control en el cual evitamos colisiones con el barco y que se dibuje en tierra.

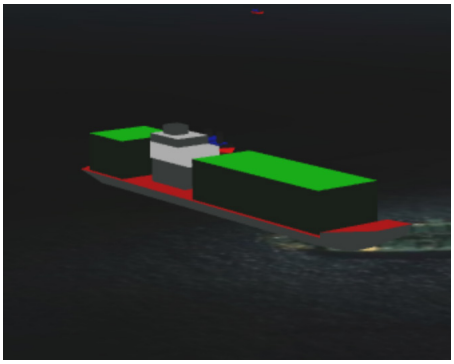


Figura 40 Modelo 3D de un barco de tipo carguero

Figura 41 Modelo 3D de un barco de tipo petrolero

Figura 42 Modelo 3D de un barco de tipo crucero

5.2.11 Sistema de alertas

El sistema de alertas, cuya arquitectura de backend fue descrita anteriormente, presenta una funcionalidad critica para la aplicación.

Se compone de varios módulos, el primero es una ventana de gestión de alertas a la cual se accede mediante la TOC.

Este gestor permite:

- Ver las alertas creadas,

- Crear nuevas alertas.

- Buscar las alertas creadas

Además incluye un pequeño mapa en dos dimensiones que nos ayuda a identificar el sensor sobre el cual creamos la alerta, apareciendo el icono del sensor en azul.

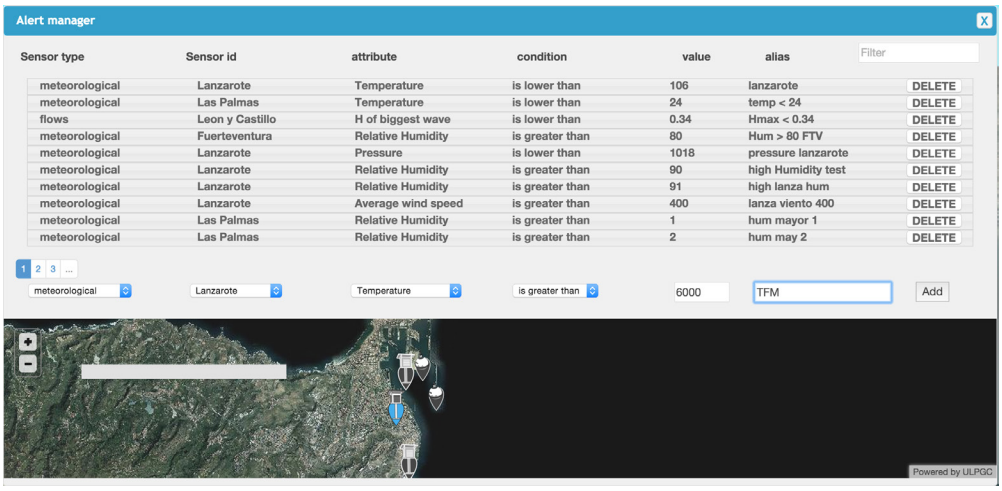


Figura 43 Pantalla de creación de alertas

Una vez creada la alerta aparece el notificador de alertas. Nos indica mediante una notificación si existe alertas existentes.

Figura 44 Notificador de alertas.

Si interactuamos con la pestaña, haciendo clic en ella, se nos despliega una ventana que nos informa la lista de alertas que han saltado. En rojo vemos las que están activas ahora mismo y las que han saltado en algún momento, aparecen en gris.

Si hacemos clic sobre una alerta se despliega una segunda venta que nos da información sobre el sensor: -

Información actual del sensor

- Foto del sensor

- Posibilidad de hacer zoom al sensor

- Información de cuando saltó la alerta

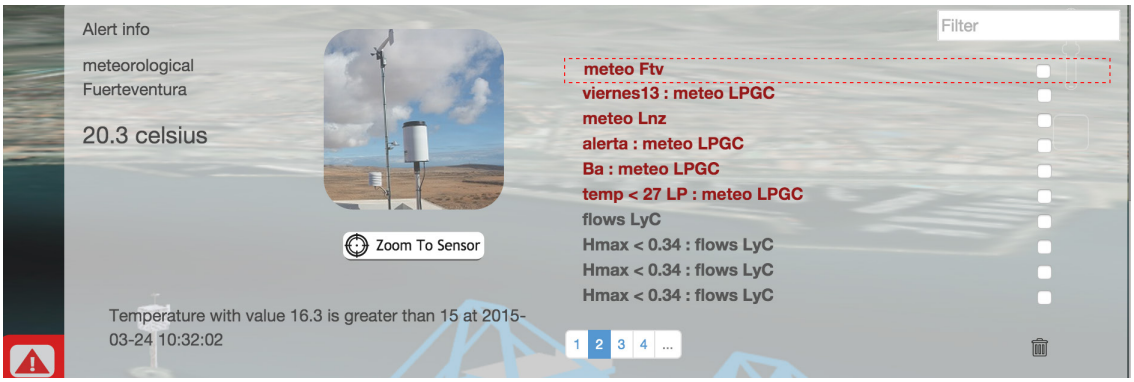


Figura 45 información de la alerta

5.2.12 Gráficas de datos

En el apartado de backend de este documento hablábamos sobre la posibilidad de capturar mediante una arquitectura de *Big Data* los datos que se han ido almacenando según los valores obtenidos de los sensores.

La visualización de dicha información la hemos mediante la librería Highcharts [60]. Esta librería nos permite generar gráficas exportables en diversos formatos (pdf, png..), lo que permiten mostrar un histórico de los datos que ofrecen los sensores el puerto.

Además tiene varias posibilidades:

- Selección según unas fechas dadas
- Al pasar el cursor sobre la gráfica, nos da la información discreta sobre, por ejemplo, un valor pico
- Selección interactiva por husos de tiempo
- Mostrar las diferentes variables que se consultan

En la guia de usuario se puede ver más detalle sobre el funcionamiento de las mismas.

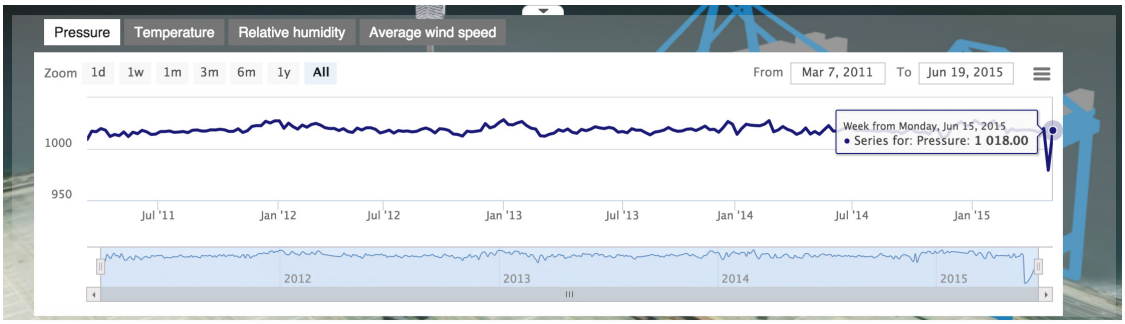


Figura 46 Datos meteorológicos desde Marzo de 2011 hasta Junio de 2015

6. Seguridad de la aplicación

Un efecto secundario del crecimiento exponencial que ha tenido el Internet es la privacidad de información tanto personal como profesional.

En Internet y por lo tanto en SmartPort es necesario protegerse frente a posibles ataques. Es importante destacar que SmartPort no es una aplicación en producción, por lo que en análisis de seguridad es algo más laxo que una aplicación que esté realmente publicada para producción. No obstante para darle cierta escalabilidad se han planteado criterios básicos de seguridad.

Mientras más se conecta el mundo, la necesidad de seguridad en los procedimientos usados para compartir la información se vuelve más importante. Desde muchos puntos de vista, podemos creer sin dudar que el punto más crítico de la seguridad del Internet, lo tienen las piezas que intervienen de forma directa con las masas de usuarios, los servidores web.

Respecto a los servidores web, hemos tenido en cuenta toda la configuración de permiso básica como (Apache, IIS, Tomcat, etc.), o en los lenguajes de programación en los que son escritas las aplicaciones que son ejecutadas por estos servidores.

Ahora que sabemos que la mayoría de los problemas de seguridad en los sitios web se encuentran a nivel aplicación y que son el resultado de escritura defectuosa de código, debemos entender que programar aplicaciones web seguras no es una tarea fácil, ya que requiere por parte del programador, no únicamente mostrar atención en cumplir con el objetivo funcional básico de la aplicación, sino una concepción general de los riesgos que puede correr la información contenida, solicitada y recibida por el sistema.

En la actualidad, aunque existen muchas publicaciones que permiten formar un criterio sobre el tema, no existen acuerdos básicos sobre lo que se debe o no se debe hacer, y lo que en algunas publicaciones se recomienda, en otras es atacado.

Por lo tanto, los elementos básicos que se han tenido en cuenta en SmartPort han sido:

— SQL Injection

— XSS

— Problemas de configuración

— Pentesting

— Gestión de sesiones y los problemas de autenticación

SQL Injection:

Descripción General

Inyección SQL es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos.

Objetivos

Se dice que existe o se produjo una inyección SQL cuando, de alguna manera, se inserta o "inyecta" código SQL invasor dentro del código SQL programado, a fin de alterar el funcionamiento normal del programa y lograr así que se ejecute la porción de código "invasor" incrustado, en la base de datos.

Este tipo de intrusión normalmente es de carácter malicioso, dañino o espía, por tanto es un problema de

seguridad informática, y debe ser tomado en cuenta por el programador de la aplicación para poder prevenirlo. Un programa elaborado con descuido, displicencia o con ignorancia del problema, podrá resultar ser vulnerable, y la seguridad del sistema (base de datos) podrá quedar eventualmente comprometida.

La intrusión ocurre durante la ejecución del programa vulnerable, ya sea, en computadores de escritorio o bien en sitios Web, en éste último caso obviamente ejecutándose en el servidor que los aloja.

La vulnerabilidad se puede producir automáticamente cuando un programa "arma descuidadamente" una sentencia SQL en tiempo de ejecución, o bien durante la fase de desarrollo, cuando el programador explicita la sentencia SQL a ejecutar en forma desprotegida. En cualquier caso, siempre que el programador necesite y haga uso de parámetros a ingresar por parte del usuario, a efectos de consultar una base de datos; ya que, justamente, dentro de los parámetros es donde se puede incorporar el código SQL intruso.

Al ejecutarse la consulta en la base de datos, el código SQL inyectado también se ejecutará y podría hacer un sinnúmero de cosas, como insertar registros, modificar o eliminar datos, autorizar accesos e, incluso, ejecutar otro tipo de código malicioso en el computador.

Por ejemplo, asumiendo que el siguiente código reside en una aplicación web y que existe un parámetro "nombreUsuario" que contiene el nombre de usuario a consultar, una inyección SQL se podría provocar de la siguiente forma:

El código SQL original y vulnerable es:

```
consulta := "SELECT * FROM usuarios WHERE nombre = '" + nombreUsuario + "';"
```

Si el operador escribe un nombre, por ejemplo "Alicia", nada anormal sucederá, la aplicación generaría una sentencia SQL similar a la siguiente, que es perfectamente correcta, en donde se seleccionarían todos los registros con el nombre "Alicia" en la base de datos:

Pero si un operador malintencionado escribe como nombre de usuario a consultar:

(sin las comillas externas), se generaría la siguiente consulta SQL, (el color verde es lo que pretende el programador, el azul es el dato, y el rojo, el código SQL inyectado):

Hay varias consideraciones que se han aplicado para mitigación de ataques de inyección SQL:

— Es importante evitar el uso de nombres de tablas dinámicas.

—Si es necesario usar los nombres de tablas dinámicas, no aceptarlos por parte del usuario, si es posible.

— Si es necesario permitir que los nombres de tablas suministradas por el usuario, utilice la funcionalidad de su sistema de gestión de base de datos para citar identificadores.

PostgreSQL, se ha usado la función quote_ident () para este fin.

Sesiones e indentificación.

Actualmente SmarPort no dispone de un sistema de identificación de usuarios. Es una de las características deseables, para entre otras cosas, se permitan tener las alertas de cada usuario.

En cuanto a la identificación, si se implementa en un futuro, se ha pensado en usar la posibilidad de logarse mediante un sistema oAuth (Open Authorization) mediante Google.

OAuth es un protocolo que permite flujos simples de autorización para sitios web o aplicaciones informáticas.

Se trata de un protocolo propuesto por Blaine Cook y Chris Messina, que permite autorización segura de una API de modo estándar y simple para aplicaciones de escritorio, móviles y web.

OAuth permite a un usuario del sitio A compartir su información en el sitio A (proveedor de servicio) con el sitio B (llamado consumidor) sin compartir toda su identidad. Para desarrolladores de consumidores, OAuth es un método de interactuar con datos protegidos y publicarlos. Para desarrolladores de proveedores de servicio, OAuth proporciona a los usuarios un acceso a sus datos al mismo tiempo que protege las credenciales de su cuenta.

De este modo evitamos una implementación compleja en la aplicación.

Cuestiones de configuración

Hemos tenido en cuenta los siguientes hitos a la hora de desarrollar la aplicación.

— Fallas de seguridad sin parchear en el software de servidor

— Defectos de software de servidor o errores de configuración que permiten la lista de directorios y ataques de recorrido de directorios

— Ajustes predeterminados innecesarios, copia de seguridad o archivos de muestra, incluyendo scripts, aplicaciones, archivos de configuración y páginas web

— Permisos de archivos y directorios impropias

— Servicios innecesarios habilitados, incluyendo la gestión de contenidos y administración remota

— Cuentas por defecto con sus contraseñas por defecto

— Las funciones administrativas o de depuración están habilitados o inaccesibles

— Mensajes de error excesivamente informativos

XSS

XSS, del inglés Cross-site scripting es un tipo de inseguridad informática o agujero de seguridad típico de las aplicaciones Web, que permite a una tercera parte inyectar en páginas web visitadas por el usuario código JavaScript o en otro lenguaje script similar (ej: VBScript), evitando medidas de control como la Política del mismo origen. Este tipo de vulnerabilidad se conoce en español con el nombre de Secuencias de órdenes en sitios cruzados.

Es posible encontrar una vulnerabilidad XSSSO en aplicaciones que tengan entre sus funciones presentar la información en un navegador web u otro contenedor de páginas web. Sin embargo, no se limita a sitios web disponibles en Internet, ya que puede haber aplicaciones locales vulnerables a XSS, o incluso el navegador en sí.

XSS es un vector de ataque que puede ser utilizado para robar información delicada, secuestrar sesiones de usuario, y comprometer el navegador, subyugando la integridad del sistema. Las vulnerabilidades XSS han existido desde los primeros días de la Web.

Esta situación es usualmente causada al no validar correctamente los datos de entrada que son usados en cierta aplicación, o no sanear la salida adecuadamente para su presentación como página web.

Esta vulnerabilidad puede estar presente de las siguientes formas:

Directa (también llamada Persistente): este tipo de XSS comúnmente filtrado, y consiste en insertar código HTML peligroso en sitios que lo permitan; incluyendo así etiquetas como <script> o <iframe>.

Indirecta (también llamada Reflejada): este tipo de XSS consiste en modificar valores que la aplicación web utiliza para pasar variables entre dos páginas, sin usar sesiones y sucede cuando hay un mensaje o una ruta en la URL del navegador, en una cookie, o cualquier otra cabecera HTTP (en algunos navegadores y aplicaciones web, esto podría extenderse al DOM del navegador).

Debido a la sencillez de SmartPort, hemos usado técnicas básicas de protección a XSS

Strip_tags es una función que nos permite limpiar cualquier etiqueta HTML que ha ingresado el usuario, de esta forma evitaremos los ataques más básicos, pero frente a un ataque más elaborado puede no ser suficiente.

PHP Input Filter es una clase desarrollada en PHP para el filtrado del posible código malicioso que se pueda insertar a través de un formulario.

Para poder emplearla hemos descargad la clase de la web oficial , incluir el archivo class.inputfilter.php al inicio y crear una instancia de la clase InputFilter

Pentesting

A pesar de lo anteriormente citado, que no es una herramienta de producción, conforme hemos ido desarrollando SmartPort hemos procurado irle aplicando algunas pruebas de pentesting automáticas.

En concreto hemos usado:

- Acunetix

- SQLMap

-nMap

7. Conclusiones y trabajo futuro

En SmartPort hemos conseguido un sistema capaz de manejar, gestionar y visualizar los datos del puerto, tanto los elementos que están geolocalizados en el territorio como los que no.

Gracias al parseo y gestión que realiza SmartPort de los datos provenientes del Puerto de Las Palmas, hemos creado una serie de funcionalidades que permite acceder a los datos de forma fluida en forma de información geográfica, paneles y gráficas.

Hemos considerado de vital importancia poder obtener un sistema con una velocidad mejorada, ya que el acceso rápido a toda la información que generan los sensores nos permite analizar, buscar patrones, y, en un futuro, prever posibles situaciones que requieran algún tipo de medida por parte del puerto.

Así mismo, gracias al gestor de alertas se ha conseguido acercarnos a un cuadro de mando geográfico; un sistema que permita, además de monitorizar y mostrar información, poder controlar puntos críticos y tomar decisiones en vivo cuando un sensor tome un valor determinado, el sistema avise al usuario para que se tomen las medidas necesarias a la hora de gestionar el puerto.

Para llevar a cabo todo esto, ha sido fundamental apoyarse en diversas tecnologías y, sobre todo, mediante combinación de las mismas hemos buscado que se de una respuesta lo más eficiente posible.

Así lo hemos hecho para poder tomar datos estáticos o dinámicos, en este último caso con soluciones de *Big Data*, tales como Hadoop integradas en el GE Cosmos e implementando una arquitectura específica. Del mismo modo ha sucedido para poder visualizar la información en 3D, que mediante Glob3Mobile, se ha podido desarrollar la parte frontal de la aplicación.

Uno de los próximos pasos de la aplicación, es tener un módulo analítico de datos. Consideramos fundamental seguir aportando inteligencia a la aplicación y poder continuar los pasos hacia un cuadro de mando geográfico para que de un salto más allá de los visores comunes de información geográfica y que SmartPort siga evolucionando en el tiempo.

Además, como se ha mencionado, la implantación de SmartPort en dipositivos móviles lo consideramos uno de los grandes hitos que se deben seguir en el proyecto. Es por ello que muchos elementos de escalabilidad fueron tenidos en cuenta en el diseño inicial de la aplicación.

Actualmente SmartPort se encuentra en un servidor de desarrollo donde hemos podido poner en prueba los conceptos y la programación realizada a lo largo del tiempo.

Cabe destacar que SmartPort ha sido una aplicación en la cual se ha invertido un esfuerzo de más de 2.000 horas de trabajo, desde la primera versión, arquitectura, implementación, testeo y corrección de errores.

8. Agradecimientos

Quiero agradecer a mi familia y amigos el constante apoyo y ayuda. Así mismo a mis tutores, Ángel Plaza y José Pablo Suárez.

Me gustaría hacer una mención especial a Agustín Trujillo que a pesar de que administrativamente no aparece como tutor del trabajo, a efectos ejecutivos ha actuado como tal.

A Conrado Domínguez, por su intermediación a la hora de conseguir los datos y su enriquecedora perspectiva de SmartPort.

Para terminar, y no por ello menos importante, al equipo que dirijo en la ULPGC, gracias a ellos SmartPort es una realidad. En orden alfabético: Sebastian Ortega, Alejando Sánchez y Jaisiel Santana.

9. Bibliografía y referencias

[1] E.F. Cood, A Relational Model of Data for Large Share Data Banks, IBM Research Laboratory, San Jose, California

[2] MySQL www.mysql.com. Último acceso Julio 2015

[3] PostgreSQL www.postgresql.org. Último acceso Julio 2015

[3] PostGIS www.postgis.net . Último acceso Julio 2015

[4] Oracle www.oracle.com. Último acceso Julio 2015

[5] Microsoft SQL Server www.microsoft.com/sql/. Último acceso Julio 2015

[6] Microsoft Access https://products.office.com/es-es/access. Último acceso Julio 2015

[7] MongoDB https://www.mongodb.org. Último acceso Julio 2015

[8] CouchDB http://couchdb.apache.org. Último acceso Julio 2015

[9] DynamoDB http://aws.amazon.com/es/dynamodb/. Último acceso Julio 2015

[10] Hbase http://hbase.apache.org. Último acceso Julio 2015

[11] Cassandra http://cassandra.apache.org. Último acceso Julio 2015

[12] Redis http://redis.io. Último acceso Julio 2015

[13] Riak http://basho.com/products/riak-kv/. Último acceso Julio 2015

[14] Neo4j http://neo4j.com. Último acceso Julio 2015

[15] OrientDB http://orientdb.com/orientdb/. Último acceso Julio 2015

[16] AeroSpike http://www.aerospike.com. Último acceso Julio 2015

[17] Hipertable http://hypertable.org. Último acceso Julio 2015

[18] Hadoop. http://hadoop.apache.org/. Último acceso Julio 2015

[19] HDFS https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. Último acceso Julio 2015

[20] The Google File System, Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung Google

[21] Map Reduce http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html. Último acceso Julio 2015

[22] Ambari https://ambari.apache.org. Último acceso Julio 2015

[23] Avro https://avro.apache.org. Último acceso Julio 2015

[24] Hbase http://hbase.apache.org. Último acceso Julio 2015

[25] Hive https://hive.apache.org. Último acceso Julio 2015

[26] Mahout http://mahout.apache.org. Último acceso Julio 2015

[27] Pig https://pig.apache.org. Último acceso Julio 2015

[28] Zookeeper https://zookeeper.apache.org. Último acceso Julio 2015

[29] Hortonworks http://hortonworks.com. Último acceso Julio 2015

[30] MapR https://www.mapr.com. Último acceso Julio 2015

[31] Cloudera http://www.cloudera.com/. Último acceso Julio 2015

[32] Bitcask https://github.com/basho/bitcask. Último acceso Julio 2015

[33] LevelDB http://leveldb.org. Último acceso Julio 2015

[34] Oracle Spatial http://www.oracle.com/es/products/database/options/spatial/index.html. Último acceso Julio 2015

[35] MySQL Spatial https://dev.mysql.com/doc/refman/5.0/en/spatial-extensions.html. Último acceso Julio 2015

[36] Microsoft SQL Server Spatial https://msdn.microsoft.com/en-us/library/bb933790.aspx. Último acceso Julio 2015

[37] GeoJinni https://locationtech.org/proposals/geojinni. Último acceso Julio 2015

[38] GeoMesa http://www.geomesa.org. Último acceso Julio 2015

[39] Cesium http://cesiumjs.org. Último acceso Julio 2015

[40] OssimPlanet http://www.ossim.org. Último acceso Julio 2015

[41] WorldWind http://worldwind.arc.nasa.gov/features.html. Último acceso Julio 2015

[42] Virtual Terrain Project http://vterrain.org. Último acceso Julio 2015

[43] Nickitas Georgas, Alan F. Blumberg, Michael S. Bruno, David S. Runnels, Marine Forecasting For The New York Urban Waters And Harbor Approaches: The Design And Automation Of NYHOPS, 3rd International Conference on Experiments/Process/System Modeling/Simulation & Optimization.

[44] Daniel E. O'leary 'Big Data', The 'Internet of Things' and the 'Internet of Signs', intelligent systems in accounting, finance and management .

[45] Stamatis Karnouskos, Thiago Nass de Holanda Simulation of a Smart Grid City with Software Agents , third UK Sim European Symposium on Computer Modeling and Simulation.

[46] Proyecto FIWARE. http://fiware.org. Último acceso Julio 2015

[47] Generic Enabler. https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FI-WAREArchitecture. Último acceso Julio 2015

[48] Glob3Mobile: https://github.com/glob3mobile/g3m. Último acceso Julio 2015

[49]Agustín Trujillo, Jose Pablo Suárez, Manuel de la Calle, Diego Gómez, A. Pedriza, J.M. Santana (2013) Glob3 Mobile: An Open Source Framework for Designing Virtual Globes on iOS and Android Mobile Devices, Progress and New Trends in 3D Geoinformation Sciences, Springer.

[50] Orion. https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Publish/Subscribe_Brok er_-_Orion_Context_Broker_-_User_and_Programmers_Guide. Último acceso Julio 2015

[51] JSON. json.org/ . Último acceso Julio 2015

[52] API REST. http://www.w3.org/2001/sw/wiki/REST. Último acceso Julio 2015

[53] NGSI 9. https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/NGSI- 9/NGSI-10_information_

model. Último acceso Julio 2015

[54] NGSI 10. https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/NGSI- 9/NGSI-10information_model. Último acceso Julio 2015

[55] AJAX. http://www.w3.org/standards/webdesign/script.html. Último acceso Julio 2015

[56] Cosmos. http://catalogue.fiware.org/enablers/bigdata-analysis-cosmos. Último acceso Julio 2015[57] Cygnus. http://catalogue.fiware.org/enablers/bigdata-analysis-cosmos

[58] Flume. http://flume.apache.org/. Último acceso Julio 2015

[59] Wirecloud http://catalogue.fiware.org/enablers/application-mashup-wirecloud. Último acceso Julio 2015

[60] Highcharts. http://www.highcharts.com/. Último acceso Julio 2015

[61] WMS. http://www.opengeospatial.org/standards/wms. Último acceso Julio 2015

[62] Makoto YUI, Isao KOJIMA. A Database-Hadoop Hybrid Approach to Scalable Machine Learning, 2013 IEEE International Congress on Big Data.

[63] McCann, Using geovrml for 3d oceanographic data visualizations. Proceedings of the ninth international conference on 3D Web technology, pages 15– 21.

[64] Villaseñor, E. and Estrada, H.) Informetric mapping of big data in fi-ware.Proceedings of the 15th Annual International Conference on Digital Government Research, pages 348–349.

[65] Leishi Zhang, Andreas Stoffel Michael Behrisch. Visual analytics for the big data era a comparative review of state-of-the-art commercial systems. Visual Analytics Science and Technology (2012) IEEE Conference on, pages 173–182. IEEE.

[66] Aviv Segev, Chihoon Jung, Sukhwan Jung. Analysis of Technology Trends Based on Big Data, IEEE International Congress on Big Data.

[67] K. Shvachko, H. Kuang, S. Radia. The hadoop distributed file system. 26th Symposium on Mass Storage Systems and Technologies, pages 1–10. IEEE.

[68] B. Yao and et al. Multi-approximation-keyword routing in gis data. 19th ACM SIGSPATIAL. Int. Conf. on Advances in Geographic Information Systems, pages 201–210.

[69] Claramunt, C., Devogele, T., Fournier, S., Noyon, V., Petit, M., and Ray, C. . Maritime GIS: from monitoring to simulation systems. Information Fusion and Geographic Information Systems, Springer. Pages 34–44.

[70] Ramparany F.,Orange Labs, Galan F, Soriano J, Elsaleh T, Handling smart environment devices, data and services at the semantic level with the FI-WARE core platform, Big Data IEEE International Conference. Calvo, M. (1992) Sistemas de Información Geográfica Digitales: Sistemas geomáticos. IVAP-EUSKOIKER, Oñati, 616 pp.

[71] Chang, K. (2007) Introduction to Geographic Information System, 4th Edition. McGraw Hill.

[72] de Smith M J, Goodchild M F, Longley P A Geospatial analysis: A comprehensive guide to principles, techniques and software tools", 2nd edition, Troubador, UK

[73] Elangovan, K "GIS: Fundamentals, Applications and Implementations", New India Publishing Agency, New Delhi"208 pp.

[74] Harvey, Francis A Primer of GIS, Fundamental geographic and cartographic concepts. The Guilford Press, 31 pp.

[75] Heywood, I., Cornelius, S., and Carver, S. An Introduction to Geographical Information Systems. Prentice

Hall. 3rd edition.

[76] Longley, P.A., Goodchild, M.F., Maguire, D.J. and Rhind, D.W. Geographic Information Systems and Science. Chichester: Wiley. 2nd edition.

[77] Maguire, D.J., Goodchild M.F., Rhind D.W. "Geographic Information Systems: principles, and applications" Longman Scientific and Technical, Harlow.

[78] Moreno Jiménez, A. (coord.); Gómez García, N., Vidal Domínguez, M. J., Rodríguez Esteban, J. A., Martínez Suárez, P., Prieto Flores, M. E., Cervera Cruañes, B. y Fernández García, F. Sistemas y análisis de la información geográfica: manual de auto-aprendizaje con ArcGIS, Madrid, Ra-Ma, 2.ª ed., 940 pp.

[79] Ott, T. and Swiaczny, F. Time-integrative GIS. Management and analysis of spatio-temporal data, Berlin / Heidelberg / New York: Springer.

[80] Olaya, V., Sistemas de Información Geográfica. Víctor Olaya. 877 pp. (Creative Common Atribución).

[81] Thurston, J., Poiker, T.K. and J. Patrick Moore. Integrated Geospatial Technologies: A Guide to GPS, GIS, and Data Logging. Hoboken, New Jersey: Wiley.

[82] Tomlin, C.Dana Geographic Information Systems and Cartographic Modelling. Prentice Hall. New Jersey.

[83] Wise, S. GIS Basics. London: Taylor & Francis.

[84] Wheatley, David and Gillings, Mark Spatial Technology and Archaeology. The Archaeological Application of GIS. London, New York, Taylor & Francis.

[85] Hrishikesh Karambelkar, Scaling Big Data with Hadoop and Solr, Packt Publixhing, ISBN 139781783281374

10. Listado de Acrónimos

ACID	Atomicity, Consistency, Isolation and Durability
AIS	Automatic Identification System
API	Application Programming Interface
CAD	Computer Aided Design
DDL	Data Definition Language
DML	Data Manipulation Language
IE	Internet Explorer
IDE	Infraestructuras de Datos Espaciales
HTTP	Hiper Text Transfer Protocol
FOSS	Free and Open-Source Software
GE	Generic Enabler
GFS	Google File System
GIS	Geographic Information System
HDFS	Hadoop Distributed File Sytem
IMO	International Maritime Organization
LOD	Level of Detail
MR	MapReduce
MVCC	Multi-Version Concurrency Control
NoSQL	Not only SQL
OGC	Open Geospatial Consortium
SIA	Sistema de Identificación Automática
SIG	Sistema de Información Geográfica
SDI	Spatial Data Infraestructure
SFSS	Simple Features Specification for SQL
SGBDR	Sistema Gestor de Bases de Datos Relacional
SQL	Structured Query Language
SSD	Solid-state Drive
TOC	Table of Contents

VBA	Visual Basic for Applications
WMS	Web Map SErvice
XAMPP	X para cualquiera de los diferentes sistemas operativos, Apache, MySQL, PHP, Perl

11. Índice de figuras y tablas

Figuras:

Tablas:

Anexo. Guía de uso de SmartPort

1. Primeros pasos.

SmartPort no se encuentra disponible mediante una url pública por lo que lo previamente nos deben haber suministrado una url válida para poder acceder a la aplicación.

Una vez accedamos, veremos la pantalla de carga. Debemos esperar hasta que termine. En este momento se está cargando el motor de G3m y los datos estáticos.

Figura 1. Pantalla de carga de SmartPort

Cuando cargue la aplicación, nos aparecerá una vista general del puerto de la luz, orientado al centro de la isla. Aquí ya podemos navegar con la aplicación. En la esquina superior izquierda nos aparecen los controles de navegación de SmartPort.

Además interactuando en la vista geográfica podemos movernos.

El panel de trabajo en SmartPort está centrado en la TOC cuyo icono vemos en la izquierda de la aplicación.

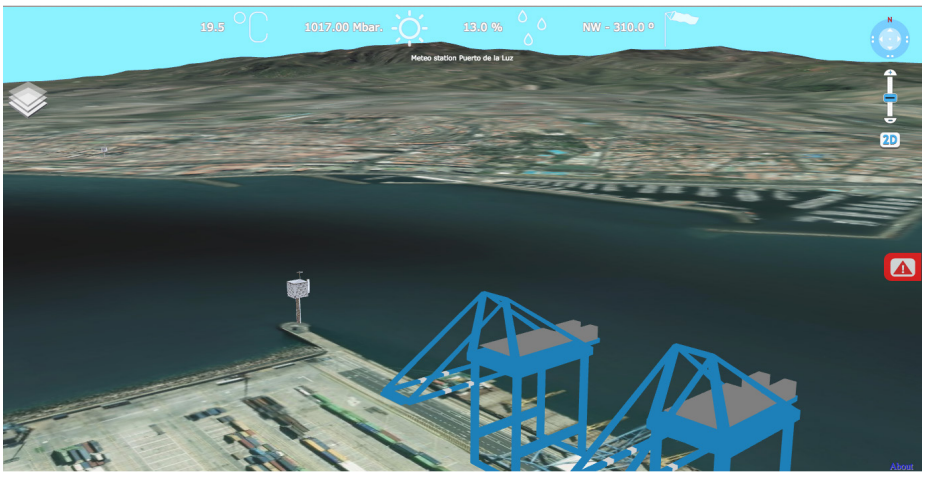


Figura 2. Pantalla principal de SmartPort. En la esquina superior derecha vemos los controles de navegación y en la izquierda el icono de la TOC.

2. La TOC

La tabla de contenidos, como su nombre indica, tiene los recursos de SmartPort.

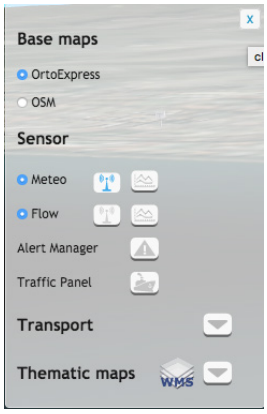


Figura 3. Tabla de contenidos

Cuando hacemos clic en ella vemos las siguientes opciones:

2.1 Base Maps:

Mapas base que se pueden usar en SmartPort. En esta versión podemos utilizar la ortoexpress de Grafcan o la capa de OpenStreetMaps

2.2 Sensor:

Los datos del sensor más cercano que se encuentre en la vista. Estos datos pueden ser de un sensor meteorológico o de una boya, pudiendo tener ambos botones desactivados. En tal caso, no aparec´er´ía ninguna información en la ventana principal.

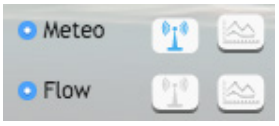


Figura 4. Iconos de la sección Sensor

Además en esta sección podemos hacer una solicitud de gráfica, esto nos daría, mediante una gráfica los datos del sensor solicitado.

Estas gráficas podemos visualizarlas seleccionando las variables disponibles, un zoom según un periodo temporal (día, semana, mes), o un rango de fechas seleccionado. Además la gráfica es interactiva, si pasamos el cursor sobre ella vemos los datos discretos.

Para terminar, en esquina superior derecha tenemos un menú para exportarla a diferentes formatos.

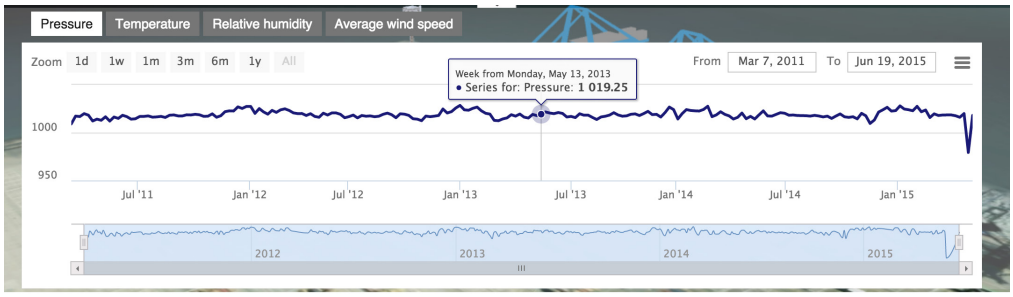


Figura 4. Gráfica de SmartPort

2.3 Alert Manager:

La funcionalidad del gestor de alertas viene determinada por el valor de los sensores. Cuando se cumpla una alerta determinada, nos aparecerá una notificación en el notificador de alertas.

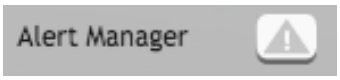


Figura 5. Alert manager en la TOC

Una vez hagamos clic en el botón que encontramos en la TOC, en el gestor de alertas podemos ver todas las alertas creadas y eliminarlas mediante el fotón DELETE. Además, mediante un cajón de texto, podemos buscar una alerta por alguna de sus palabras clave.

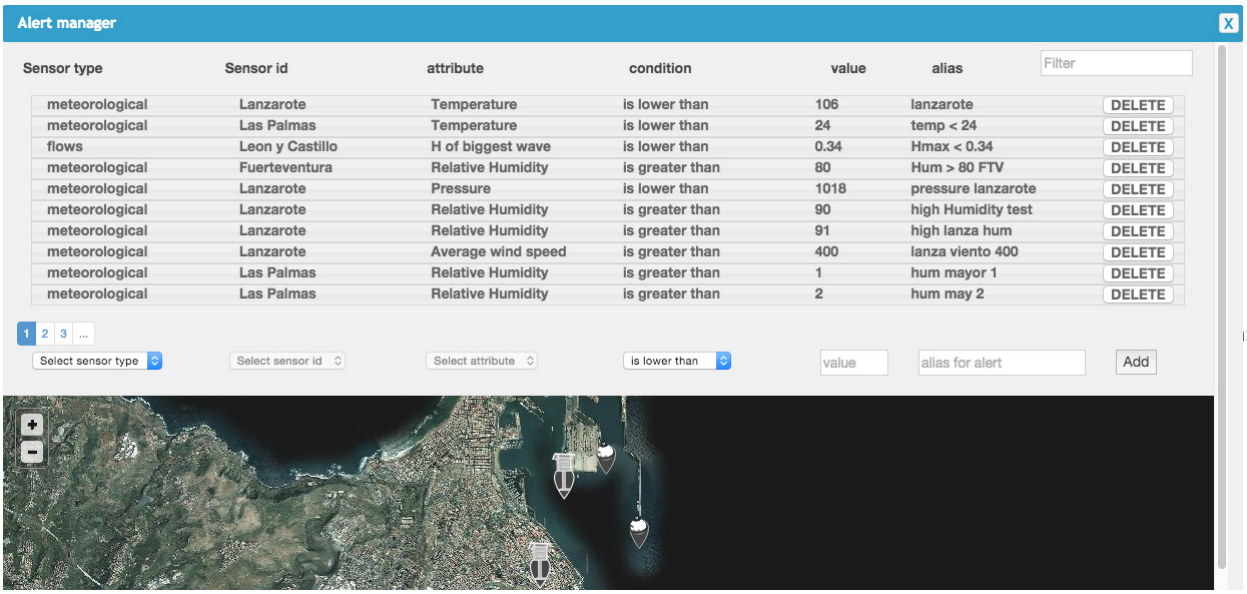


Figura 6. Alert manager

- Tipo de sensor: Meteorológico o boyas

- Sensor: tomará el valor de los sensores disponibles de cada tipo. Una vez seleccionados los dos primeros items, en el mapa del gestor de alertas aparecerá el sensor en azul para poder identificar su localización geográfica.

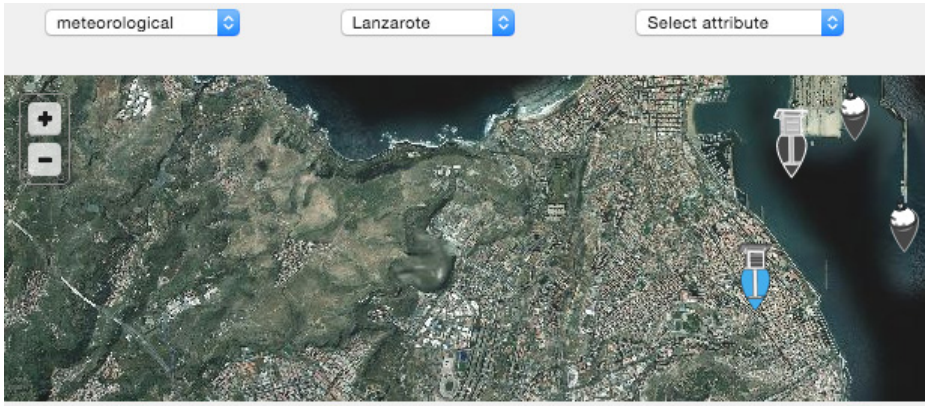


Figura 7. Resaltado de sensor en el mapa del Alert Manager

- Atributo sobre el cual se quiere hacer la alerta. Por ejemplo, en meteorológico puede ser sobre sus atributos tales como temperatura, humedad, presión, etc.

- Condición: si está por debajo, es igual o por encima del valor que indicaremos en el siguiente selector.

- Valor: sobre el cual crearemos la alerta

- Alias: para identificar la alerta y poder reconocerla y/o buscarla.

- Botón de añadir. Para crear la alerta

2.4 Traffic panel

El panel de tráfico en SmartPort pretende dar la información diaria de las entradas y salidas de barcos en el puerto de la Luz.

La idea originalmente surgió del ganador de un hackathlon del puerto de Singapur, siendo la aplicación ganadora un panel tipo aeropuerto.

Haciendo clic en el botón correspondiente saldrá la ventana de información de tráfico de barcos.

Figura 8.Traffic Panel en la TOC

Figura 9. Traffic Panel desplegado

Si deseamos conocer más detalles sobre una llegada/Salida de barcos, haciendo clic en el nombre del barco obtendremos más información detallada sobre el mismo:

-id

- Longitud

- Nacionalidad

- Dique

- Compañía

- Origen

- Destino

Además se dispone de una caja de búsqueda por palabras clave.

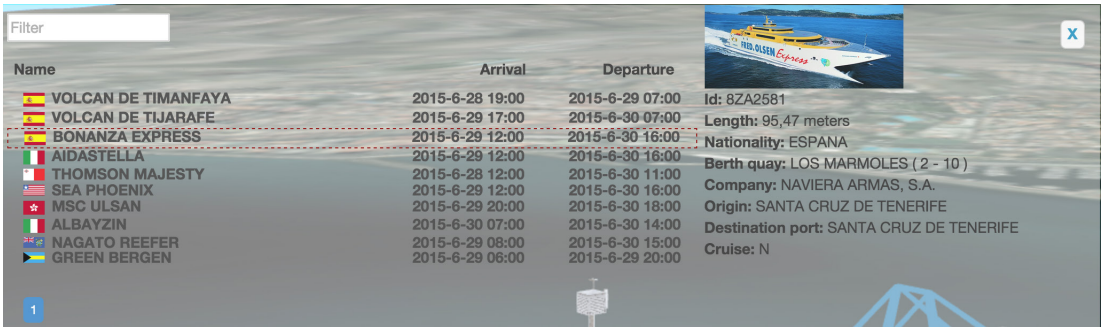


Figura 10. Traffic Panel desplegado con información de un barco

2.5 Transport

Transport nos renderizará en el visor geográfico los modelos en tres dimensiones de los barcos según la última ubicación conocida. Esto difiere con el traffic panel en que la información proviene del AIS, por lo que se tiene en cuenta tanto las grandes entradas/salidas así como otras de menos importancia.

Par ver todas las opciones disponibles debemos hacer clic en la flecha que se encuentra al lado del texto Transport

Figura 11. Transport en la TOC

Una vez hagamos clic en ella veremos todas las opciones disponibles:

- Cruceros y barcos de cruceros

- Cargueros

-Tanques

-Vela y barcos de placer

-Barcos de alta velocidad

- Otros barcos

- Grúas (logísitica)

- Contenedores (logística)

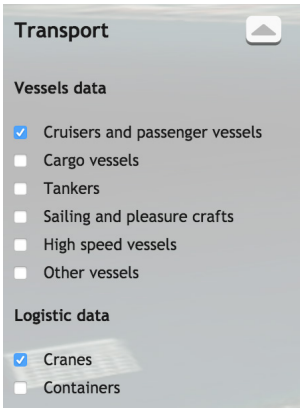


Figura 12. Transport panel desplegado. En este caso se muestra la información de cruceros y grúas.

Al activar y desactivar cada uno de estos elementos aparecerán los modelos para cada categoría, si hay en ese momento alguno en el puerto.

Además cada modelo es interactivo; haciendo clic en él obtendremos toda la información que nos aporta el AIS. En el caso de las grúas no aportan información y los contendores aportan información no real a modo de demo.

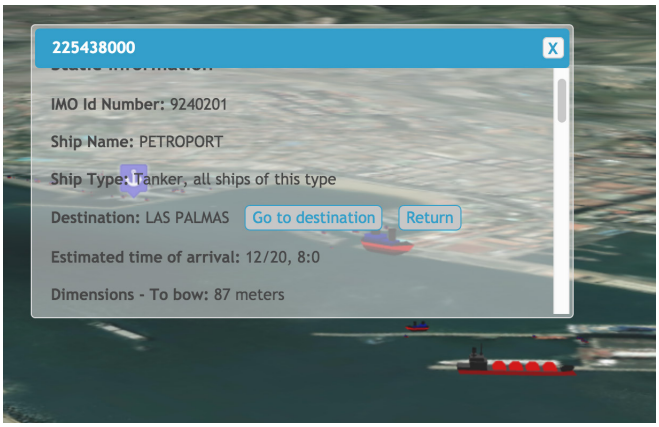


Figura 13. Información de un barco.

Dentro de la ventana de información del AIS podemos interactuar además con dos botones:

-Go to destination: el visor se situará en el puerto destino si se encuentra en la base de datos

-Return: para volver a la vista anterior

2.6 Thematic info

Thematic info nos da información sobre todas las capas temáticas disponibles por defecto y además nos deja cargar nuevas mediante fuentes de datos WMS.

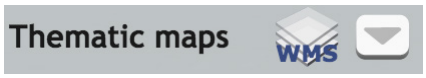


Figura 14. Thematic maps en la TOC

Si hacemos clic en la flecha que va hacia abajo, veremos todas las capas disponibles.

Por defecto, como servicios WMS tenemos:

- Batimetría

- Espacios naturales

Y como datos estáticos tenemos:

- Acomodación

- Restaurantes

- Cajeros automáticos

- Paradas de guaguas

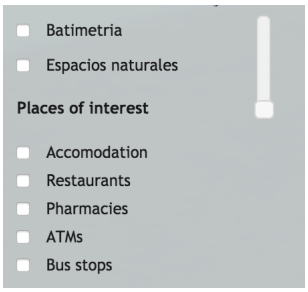


Figura 15. Thematic maps expandido

A las dos primeras capas podemos añadirle transparencia, para ello debemos seleccionarlas y la barra se activará, pudiendo moverla para aplicar la transparencia deseada.

Figura 16. Aplicando transparencia a espacios naturales

Además, haciendo clic en el botón WMS podemos cargar servicios interoperables, los pasos son los siguientes:

1.- Hacemos clic en WMS

Figura 17. Botón de WMS en Thematic maps

Nos aparece un diálogo donde ponemos la url del servicio, en este caso usaremos el servicio de GrafCan de información estadística:

http://idecan2.grafcan.es/ServicioWMS/CARTO_EST?

2.- Hacemos clic en Load.

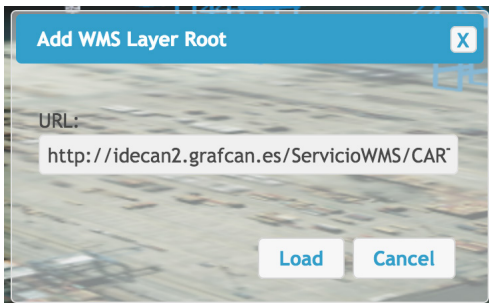


Figura 18. Ventana donde se pone la url del WMS.

3.- Nos debe cargar todas las capas disponibles, seleccionamos las que queramos añadir al visor, y podemos cambiar el nombre que aparecerá en la TOC si lo deseamos.

4.- Hacemos clic en load

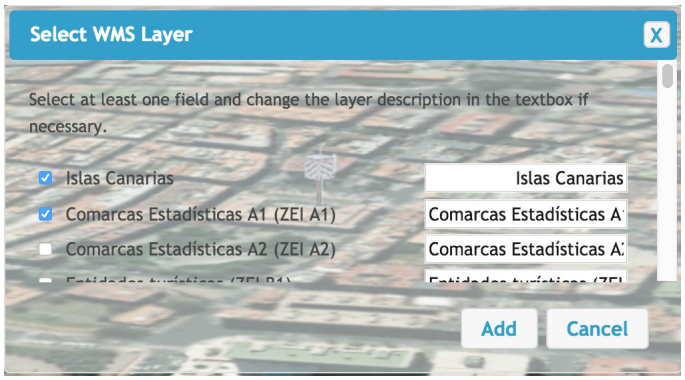


Figura 19. Capas disponibles. En este caso cargaríamos "Islas Canarias" y "Comarcas Estadísticas A1"

5.- Ya las capas deberían estar en la TOC

3 Notificador de alertas

Anteriormente, en la TOC creamos y gestionamos alertas. Estas alertas se presentan a con un notificador que aparece a la derecha del visor, si existe alguna activa.

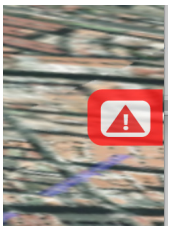


Figura 20. Notificador de alertas

Si hacemos clic en él, nos apareceran todas las alertas que han saltado en algún momento y las activas. Las activas (alertas que se están cumpliendo en este momento) aparecerán en rojo, mientras las las que han saltado en algún momento, aparecerán en gris.

Además, tenemos una caja de texto que nos permitirá buscar las alertas por palabras clave.

Unido a eso, tenemos un check a la izquierda, donde podemos seleccionar la notificación de la alerta para borrarla haciendo clic en la papelera que se encuentra en la parte inferior de la ventana

Figura 21. Notificador de con alertas activas y no activas.

Si hacemos clic en una alerta podemos ver detalles de la misma:

- Sensor

- Valor actual del sensor para la variable

- Foto del sensor

- Botón para hacer zoom a ese sensor

- Información sobre cuando saltó la alerta

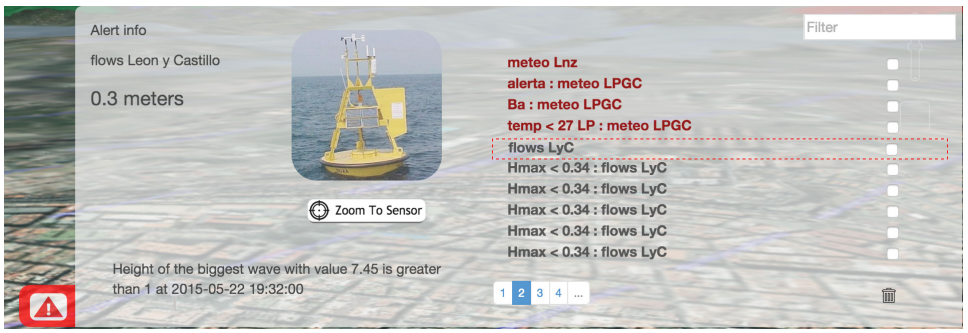


Figura 22. Información expandida de la alerta. en este caso de temp < 27 en meteo LPGC

4. About

Finalmente, haciendo clic en la esquina inferior derecha del visor podemos ver la información relativa al proyecto.

Figura 23. Información del about.

Máster IUMA

ULPGC

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA