



## Proyecto SORGES

***SOFTWARE DE REPRESENTACIÓN GRÁFICA DE EVENTOS SÍSMICOS***

Adán Toscano López  
José Felipe Sánchez Arenas  
Rubén de Jesús Moreno Leyva  
Andres Sanchez Anillo

GRADO EN INGENIERÍA INFORMÁTICA  
ESCUELA SUPERIOR DE INGENIERÍA  
UNIVERSIDAD DE CÁDIZ

26 de Enero de 2015

## Licencia

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

## Índice General

- 1. Introducción**
  - 1.1. Motivación**
  - 1.2. Objetivos**
  - 1.3. Alcance**
  - 1.4. Definiciones, acrónimos y abreviaturas**
  - 1.5. Estructura del documento**
- 2. Planificación**
  - 2.1. Metodología del desarrollo**
  - 2.2. Planificación del proyecto**
  - 2.3. Organización**
  - 2.4. Costes**
  - 2.5. Gestión de riesgos**
- 3. Análisis de requisitos**
  - 3.1. Especificación de los requisitos funcionales**
  - 3.2. Especificación de los requisitos no funcionales**
  - 3.3. Especificación de los requisitos de rendimiento**
  - 3.4. Actores del proyecto**
  - 3.5. Estudio de tecnologías alternativas**
  - 3.6. Análisis GAP**
- 4. Diseño**
  - 4.1. Diseño conceptual**
  - 4.2. Diseño lógico**
  - 4.3. Diseño físico**
- 5. Implementación**
  - 5.1. Entorno tecnológico**
  - 5.2. Código fuente**
- 6. Pruebas**
  - 6.1. Pruebas unitarias**
  - 6.2. Integración**
  - 6.3. Prueba del sistema**
- 7. Conclusiones**
  - 7.1. Aspectos generales**
  - 7.2. Conocimientos adquiridos**
  - 7.3. Futuro del proyecto**
- 8. Referencias y Bibliografía**

## **ANEXOS**

- A. Herramientas**
- B. Publicación de la aplicación**
- C. Manual de instalación**
- D. Manual de usuario**
- E. Licencia**

# 1. Introducción

## 1.1. Motivación

El proyecto **SORGES** (*software de representación gráfica de eventos sísmicos*) nace con el objetivo de desarrollar un sistema software de tiempo real que sirva para mostrar gráficamente en un mapa geográfico la actividad sísmica recogida por la red de estaciones receptoras que controla el **Real Observatorio de la Armada** de la ciudad de San Fernando (Cádiz). El sistema tiene la intención de ser una extensión a la funcionalidad implementada por el módulo **ALERT-ES**, un sistema de alerta temprana (detección casi en tiempo real de eventos sísmicos) implementado en un proyecto, en el cual el Observatorio ha participado, que a su vez buscaba aumentar la funcionalidad proporcionada por el sistema **Seiscomp3**, un software alemán de adquisición, procesamiento, distribución y interactiva análisis.

SORGES es un proyecto desarrollado por un equipo de alumnos de la universidad de Cádiz, dentro de la asignatura *Proyectos Informáticos* del último curso del Grado en Ingeniería Informática, ofreciendo la capacidad al proyecto *ALERT-ES* de disponer de la opción de representar sus datos gráficamente en una imagen de mapa, complementando así la funcionalidad de Seiscomp3 ya que este no representa el estado de las estaciones ni la

SORGES contiene además del sistema de representación gráfica, un módulo almacenamiento del historial, capacitando al sistema de almacenar todas las etapas de un evento, de la misma manera que se representó por primera vez en el sistema.

El sistema actualmente en uso en las instalaciones del Real Observatorio consta de dos subsistemas fundamentalmente, el sistema Seiscomp3 y el sistema Alertes:

- **Seiscomp3:**

El SeisComP software sobre eventos sísmicos ha evolucionado notablemente a partir de módulos software de monitoreo de terremotos en tiempo real con gran cantidad de funciones. El protocolo SeedLink ahora se ha dotado de gran popularidad para la transmisión de datos sísmicos, este protocolo fue el núcleo de SeisComP desde sus inicios. Las Mejoras posteriores dotaron al software de capacidades de detección de eventos, ubicación y determinación magnitud simples, de forma automática. La comunicación entre los módulos se consigue utilizando una infraestructura TCP/IP, permitiendo en el sistema la computación y la revisión remota distribuida.



Características generales del sistema:

- SEEDLINK para la adquisición de datos en tiempo real

- ArcLink para el acceso rápido a los datos de archivo
- C ++ bibliotecas encarecidamente para el acceso de base de datos, actualmente con soporte para MySQL
- Cálculos sismológicos
- Cálculos de ubicación
- Cálculo magnitud
- Las interfaces gráficas de usuario (GUI) para
- QuakeML importación / exportación
- Contenedores para el lenguaje de programación Python
- SeisComP 3 se ejecuta actualmente en la mayoría de los sistemas operativos Unix o Unix, incl.
- Compatible con Linux (todas las versiones razonablemente recientes de Ubuntu, SuSE, etc.), Solaris y MacOSX (experimental!).
- Actualmente no es compatible con Windows.

- **ALERT-ES:**

ALERT-ES (Sistema de Alerta Sísmica Temprana: Aplicación al Sur de España) es un proyecto de investigación coordinado entre la Universidad Complutense de Madrid (UCM, Subproyecto 1, I.P. Dra. E. Buforn), el Real Instituto y Observatorio de la Armada (ROA, Subproyecto 2, I.P. Dr. J. Martín Dávila) y el Instituto Geológico de Cataluña (IGC, Subproyecto 3, I.P. Dr. X. Goula), actuando como coordinadora la Dra. E. Buforn.



En el proyecto ALERT-ES se propone estudiar la viabilidad de un Sistema de Alerta Sísmica Temprana (SAST) para los terremotos potencialmente destructores que ocurren en la zona del Cabo S. Vicente-Golfo de Cádiz. La aplicación de la alerta sísmica temprana al caso de terremotos (**SATS**) es similar: la alerta se da tras detectarse en una red sísmica un terremoto con posibles efectos destructores antes de que éstos se produzcan en emplazamientos más lejanos y siempre que cumpla unos determinados requisitos. Los SAST están basados, por tanto, en modernos sistemas de información sísmica en tiempo real que permite dar una notificación rápida de los efectos potencialmente destructores de un terremoto, permitiendo

mitigar algunos de los efectos devastadores del mismo. Los Sistemas de Alerta Sísmica Temprana (SAST) son, por tanto, sistemas de alarma ante la ocurrencia de un terremoto destructor.

Los objetivos generales del proyecto es el estudio de la viabilidad de un sistema SAST para los terremotos del Cabo S. Vicente- Golfo de Cádiz, y el desarrollo de algoritmos para la estimación rápida de la magnitud de los terremotos del sur de España a partir del estudio de los primeros segundos de la onda P.

## 1.2. Objetivos

### **OBJ-001: Representación gráfica de eventos sísmicos:**

El objetivo general del proyecto es la representación gráfica de eventos sísmicos que acontecen en la Bahía de Cádiz a través de los datos que nos suministra el sistema Seiscomp3 y Alert-es.

A partir de este objetivo, derivan los siguientes subobjetivos:

- **OBJ-002: Representación tanto en tiempo real como en simulación** con datos ya registrados **de eventos sísmicos** (llamados **ORÍGENES Y EVENTOS**, siendo los eventos la última manifestación de los orígenes) procedentes del sistema *Alert-es* en un mapa geográfico, así como de la **expansión de la onda S** (que es la onda destructiva de los seísmos).
- **OBJ-003: Representar las estaciones** involucradas en la alerta con su correspondiente color dependiendo del grado de destrucción del evento sísmico.
- **OBJ-004: Mostrar junto al mapa los datos relacionados** con los orígenes/eventos y estaciones presentes en el mismo mapa.
- **OBJ-005: Almacenar un histórico** de eventos recibidos por el sistema, para su estudio y reproducción en el futuro.

## 1.3. Alcance

En cuanto a las consideraciones sobre el alcance del proyecto, se debe tener en cuenta:

- La zona representada por el sistema se delimitará por las coordenadas 14° a 3° longitud oeste y 34° a 40° latitud norte.
- La expansión de la onda sísmica se tendrá como constante la media de 3500 metros por segundo, independientemente de la superficie de expansión.

## 1.4. Definiciones, acrónimos y abreviaturas

- **SORGES:** software de representación gráfica de eventos sísmicos.
- **ALERT-ES:** Sistema de Alerta Sísmica Temprana: Aplicación al Sur de España.
- **Pick:** Primera pista del evento (seísmo). Es el primer pulso con una amplitud fuera de lo normal que recibiría cada estación sísmica. Puede haber picks falsos por causas diversas pero a nuestro sistema solo van a llegar los reales.
- **Alerta on-site:** Código de colores (verde, amarillo, naranja y rojo) que indica el nivel de alerta de las estaciones.
- **Magnitud:** Valor en la escala Richter del origen/evento.
- **Origen:** Cada uno de los estados de la actividad que se van recibiendo desde las estaciones, sucesivos orígenes irán llegando más actualizados y con más información. El último origen es ya el evento (seísmo) en sí.
- **SAST:** Sistema de Alerta Sísmica Temprana.
- **Onda S:** Onda destructiva del seísmos.
- **XML:** (Extensible Markup Language), lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

## 1.5. Estructura del documento

Este documento está compuesto por las siguientes partes:

- Introducción: pequeña descripción del proyecto, así como los objetivos y estructura del documento.
- Descripción general: descripción más amplia sobre el proyecto, así como todas las características relevantes que tendrá.
- Planificación: exposición de la planificación del proyecto y las distintas etapas que esta compuesto el mismo.
- Análisis: fase de análisis del sistema, empleando la metodología seleccionada. Se definirán los requisitos funcionales del sistema, diagramas de caso de uso, diagramas de secuencia y contrato de las operaciones.
- Diseño: realización del diseño del sistema, diagramas de secuencia y clases aplicadas al diseño.
- Implementación: aspectos más relevantes durante la implementación del proyecto. Y problemas que han aparecido durante el desarrollo de este.
- Pruebas y validaciones: pruebas realizadas a la aplicación, con el fin de comprobar su correcto funcionamiento y cumplimiento de las expectativas.
- Conclusiones: conclusiones obtenidas tras el desarrollo de la aplicación.
- Bibliografía: libros y referencias consultadas durante el desarrollo del proyecto.
- Apéndices:

- Herramientas utilizadas: explicación de todas las herramientas usadas a lo largo del desarrollo del proyecto.
- Manual de instalación: manual para la correcta instalación del proyecto en el sistema.
- Manual de usuario: manual de usuario para el correcto uso de la aplicación.
- Licencia GNU GFDL: texto completo sobre la licencia GNU GFDL en inglés.

## 2. Planificación

### 2.1. Metodología del desarrollo

Para este proyecto vamos a usar un enfoque de desarrollo rápido de aplicaciones. Este modelo se basa en la creación de un software en el menor tiempo posible y con la mayor calidad en la entrega.

Aunque esta metodología hace especial hincapié al aspecto comercial, no lo vamos a tener en cuenta ya que nuestro software no va a ser comercial. Salvo este punto los demás principios de esta metodología se adecuan perfectamente a nuestras necesidades.

A continuación se enumeran los principios básicos de este modelo:

- Partir el proyecto en segmentos más pequeños para proporcionar facilidad de cambio durante el proceso de desarrollo y así reducir riesgos.
- Creación de prototipos en cualquier fase del desarrollo para la depuración del mismo.
- Se hace especial hincapié en el cumplimiento de los plazos de entrega, se reduce los requisitos antes de aplazar la fecha.
- Los componentes se encuentran en constante participación con el proyecto.
- No se centra en el desarrollo de un único prototipo.
- La documentación describe todo lo necesario para un futuro desarrollo y mantenimiento.

Esta metodología se adapta a los recursos disponibles, cada integrante del grupo del proyecto se va a dedicar a un módulo, al que se le aplicarán pruebas durante el desarrollo, para finalmente integrarlos todos y formar un prototipo y del cual podremos seguir modificando por módulos hasta llegar al producto final.

Hemos elegido este tipo de metodología frente a otras como pueden ser la de cascada, prototipado o espiral ya que nuestro trabajo no podía estar centrado en una sola línea de desarrollo(cascada) ni a un único prototipo, debido a que enfocar todo nuestro trabajo a cada parte del proyecto no sería eficiente y no se cumplirían los plazos. Tampoco hemos elegido el

modelo en espiral a causa del poco tiempo que abarca el proyecto, no podemos hacer un ciclo por cada módulo del proyecto.

## 2.2. Planificación del proyecto

Nombre	Fecha de inicio	Fecha de fin	Duración
Reunión con el cliente	5/11/14	5/11/14	0
Investigación	5/11/14	11/11/14	7
Planificación del desarrollo, herramientas y repositorio del proyecto	5/11/14	11/11/14	7
Elicitación y análisis de requisitos	5/11/14	11/11/14	7
Confirmación de requisitos	12/11/14	12/11/14	0
Desarrollo	14/11/14	30/12/14	47
Diseño lógico de modelo de datos	14/11/14	20/11/14	7
Preparación del entorno de desarrollo	14/11/14	20/11/14	7
Realización de módulos	21/11/14	17/12/14	27
Implementación de prototipo	18/12/14	30/12/14	13
Prototipo	31/12/14	31/01/15	32
Pruebas de prototipo	31/12/14	9/01/15	10
Prototipo funcional	10/01/15	10/01/15	0
Mostrar prototipo al cliente	10/01/15	10/01/15	0
Correcciones de diseño e implementación	10/01/15	24/01/15	15
Pruebas para depuración de fallos	25/01/15	31/01/15	7
Versión final	1/02/15	1/02/15	0
Documentación	5/11/14	31/01/15	88



## **2.3. Organización**

El proyecto está organizado por fases, mostradas en el anterior apartado, en las que se puede destacar tres fases distintas.

En primer lugar nos citamos con el cliente y buscamos las mejores opciones para satisfacer las necesidades del proyecto durante una semana, para más tarde reunirnos y discutir las opciones. De esta forma dejamos claro las herramientas necesarias y realizamos una estructura para la ejecución del proyecto.

Tras dejar claro los módulos del proyecto, comenzamos a implementarlos. Los integrantes del proyecto nos repartimos las tareas de tal forma:

- Adán Toscano López (Jefe del Proyecto): Organización, contacto con el cliente, supervisión de pruebas y documentación.
- José Felipe Sánchez Arenas (Desarrollador): Entorno gráfico.
- Rubén de Jesús Moreno Leyva (Desarrollador): procesamiento de datos.
- Andres Sanchez Anillo (Desarrollador): Procesamiento de datos.

Aunque la división sea en principio la indicada, la envergadura y naturaleza del proyecto hace que las obligaciones acaben siendo transversales y cada miembro del grupo ha intervenido al menos en una parte de todas las actividades divididas.

Una vez a la semana ponemos una puesta en común presencial, para resolver dudas y problemas que puedan surgir y conocer todos los módulos que constituyen el proyecto.

Una vez implementado los modulos, los integramos para crear un prototipo. Con este prototipo vamos resolviendo los problemas que puedan surgir e implementando nuevas funcionalidades hasta llegar al producto final.

Cabe destacar que se han usado herramientas como *Dropbox*, *GitHub*, *Drive*, *Gmail* y *Whatsapp* para estar en continuo contacto durante el desarrollo del proyecto.

## **2.4. Costes**

Todas las herramientas usadas son gratuitas con lo que no se reflejan costes en software. El mayor peso de los costes recae en los recursos humanos, los cuales son, las horas de investigación, diseño, realización e implementación del proyecto el cual se realiza en 3 meses, con una dedicación de 8 horas semanales por persona.

Otra parte del coste se asigna a los gastos indirectos como luz, transporte, etc.

- Coste de personal:

Tarea	Horas	Costes(€)
Análisis del problema	5	150
Documentación	85	2.550
Diseño de la Aplicación	82	2.460€
Implementación de la Aplicación	68	2.040€
Evaluación de la Aplicación	55	1.650€
Redacción de la memoria	25	750€
<b>Total</b>	320	9.600€

El cálculo realizado de los costes del personal son por cada personal involucrado en el proyecto.

- Coste material:

Equipamiento	Cantidad	Costes
Portátiles de trabajo y simulación	4	2.000€
Gastos eléctrico	-	300€
Gastos de transporte	-	100€
<b>Total</b>	-	2.400€

- Precio Estimado:

Tipo	Coste	Total
Humano	600€/mes por persona	7.200€
Indirecto	800€/mes	2.400€
	<b>Total</b>	9.600€

## 2.5. Gestión de riesgos

Documentación de información de riesgos			
ID del riesgo:	RS-01	Fecha:	24/12/2014
Prob:	90%	Impacto:	Muy Alto
<p><b>Descripción:</b> Retrasos en la especificación. La información sobre las especificaciones de datos y formato de ficheros proporcionada por los técnicos del Real Observatorio de la Armada, se retrase poniendo en peligro la integridad del proyecto.</p>			
<p><b>Contexto:</b></p> <ul style="list-style-type: none"><li>Contexto 1: El equipo técnico encargado en el Real Observatorio de la Armada nos proporciona tarde los datos para realizar la conexión mediante socket, y finalmente se anula este requisito.</li><li>Contexto 2: El equipo técnico encargado en el Real Observatorio de la Armada nos proporciona tarde la información relacionada con el formato del fichero de entrada.</li></ul>			
<p><b>Reducción:</b></p> <ul style="list-style-type: none"><li>Insistir en el proporcionamiento de información respecto a la salida del sistema al que debemos integrar el proyecto.</li><li>Realizar los modulos no relacionados con los modulos de Socket y procesamiento de datos de entrada.</li><li>Desarrollar un sistema de procesamiento de datos extrayendo estos directamente de los logs del sistema ALERT-ES aunque no sigan formato ninguno.</li></ul>			
<p><b>Plan de contingencia:</b></p> <ul style="list-style-type: none"><li>Realizar un programa independiente del programa que simule la conexión mediante sockets, para prescindir del sistema.</li><li>Cambiar el formato de entrada a formato XML, y posteriormente realizar una conversión del formato proporcionado por el cliente a XML.</li><li>Orientar el procesamiento de datos de la aplicación directamente a la lectura de ficheros en una ruta determinada.</li></ul>			
<p><b>Actual estado:</b> 12/Enero/2015 El proyecto por orden del cliente se reduce y no obtenemos los datos de entrada mediante comunicación por Sockets, y se procesará la información directamente de los ficheros de log generados por los sistemas ALERT-ES y los ficheros xml de SEISCOMP3.</p>			
<b>Desarrollado:</b> Adan Toscano López	<b>Asignado a:</b>		

	<ul style="list-style-type: none"> <li>• Rubén Moreno Leyva</li> <li>• Felipe Sánchez Arena</li> <li>• Andrés Sánchez Anillo</li> <li>• Adán Toscano López</li> </ul>
--	---

Documentación de información de riesgos			
ID del riesgo:	RS-03	Fecha:	24/11/2014
Prob:	85%	Impacto:	Muy alto
<b>Descripción:</b> Cambio de requisitos. Los técnicos del Real Observatorio de la Armada consideren pertinente el cambio de algún requisito del proyecto en medio de su desarrollo.			
<b>Contexto:</b> <ul style="list-style-type: none"> <li>• Contexto 1: Los técnicos encargados de dar las indicaciones para el desarrollo del proyecto, eliminan o modifiquen algún módulo predefinido, con otro módulo dependiente.</li> </ul>			
<b>Reducción:</b> <ul style="list-style-type: none"> <li>• Realizar reuniones y definir claramente los requisitos, y definir posibles módulos con futuras modificaciones.</li> <li>• Desarrollar modulos dependiente de otros módulos genéricos, reduciendo el impacto si surgiera un cambio él o el módulo al que depende.</li> <li>• Tener contacto constante con el cliente, para obtener los cambios de requisitos con el mayor tiempo posible.</li> </ul>			
<b>Plan de contingencia:</b> <ul style="list-style-type: none"> <li>• Reorganizar los grupos de desarrollos para el apoyo a aquellos módulos afectados por el cambio de requisitos.</li> </ul>			
<b>Actual estado:</b> 20/Enero/2015 El proyecto ha sufrido cambios de requisitos, eliminando el módulo de conexión Socket, gracias a la gestión del riesgo RS-03, el impacto ha sido mínima, pudiendo utilizar el módulo dependiente de procesamiento de datos, sin necesidad de tener el módulo eliminado.			
<b>Desarrollado:</b> Adan Toscano López	<b>Asignado a:</b> <ul style="list-style-type: none"> <li>• Rubén Moreno Leyva</li> <li>• Felipe Sánchez Arena</li> </ul>		

Documentación de información de riesgos			
ID del riesgo:	RS-02	Fecha:	23/11/2014
Prob:	70%	Impacto:	Alto
<b>Descripción:</b> Subestimar el tamaño. Con la información inicial proporcionada por el personal técnico del Real Observatorio de la Armada, el equipo de organización dedicado a SORGES, estime un tiempo menor en el desarrollo de algunos de los modulos que componen el proyecto.			
<p><b>Contexto:</b></p> <ul style="list-style-type: none"> <li>Contexto 1: La organización del proyecto estima que el tiempo en el desarrollo del módulo de procesamiento de ficheros de entradas es menor al finalmente empleado.</li> <li>Contexto 2: La organización del proyecto estima que el tiempo en el desarrollo del módulo de representación gráfica es menor al finalmente empleado.</li> <li>Contexto 3: El equipo de desarrollo dedicado en la integración de cada módulo en el proyecto, se encuentren problemas para integrar algún módulo por incompatibilidades entre módulos.</li> </ul>			
<p><b>Reducción:</b></p> <ul style="list-style-type: none"> <li>Obtener información general al respecto de cada módulo y tener una idea general de cómo realizaremos los módulos y las principales herramientas que utilizaremos.</li> <li>Nombrar a un subequipo de desarrolladores encargados de implementar cada submódulo que se van finalizando, y modificaciones de cada módulo.</li> <li>Predefinir 12 días al final para integrar todos los módulos en el proyecto.</li> </ul>			
<p><b>Plan de contingencia:</b></p> <ul style="list-style-type: none"> <li>Reorganizar los grupos de desarrollos para el apoyo a aquellos módulos con retraso en la fecha de finalización, o estimada finalización.</li> <li>Reorganizar los grupos de desarrollos para el apoyo a la integración de los módulos finalizados por integrar.</li> </ul>			
<p><b>Actual estado:</b> 14/Enero/2015 El proyecto actualmente tiene el módulo gráfico planteado, e integrado en el proyecto a esperas de posibles cambios en el sistema con el procesamiento de datos.</p>			
<b>Desarrollado:</b> Adan Toscano López		<b>Asignado a:</b> <ul style="list-style-type: none"> <li>Rubén Moreno Leyva</li> <li>Felipe Sánchez Arena</li> <li>Andrés Sánchez Anillo</li> </ul>	

Documentación de información de riesgos			
ID del riesgo:	RS-06	Fecha:	10/12/2014
Prob:	40%	Impacto:	Alto
<b>Descripción:</b> Rendimiento de las herramientas. En el proceso de desarrollo el equipo no recibe todo el rendimiento ni las características que esperaban de la tecnología utilizada para el proyecto.			
<b>Contexto:</b> <ul style="list-style-type: none"> <li>Contexto 1: La biblioteca Qt utilizada por el equipo de desarrollo del proyecto, imposibilita la realización de procesamiento requerido a tiempo real.</li> <li>Contexto 2: La biblioteca Qt utilizada no nos permite representar gráficamente las características que se definieron en la reunión con el cliente.</li> <li>Contexto 3: La biblioteca Qt utilizada no nos permite procesar de forma eficiente los datos de un fichero.</li> </ul>			
<b>Reducción:</b> <ul style="list-style-type: none"> <li>Realizar informe sobre la tecnología que vamos utilizar y tecnologías alternativas al sistema.</li> </ul>			
<b>Plan de contingencia:</b> <ul style="list-style-type: none"> <li>Conservar los algoritmos utilizados y reorganizar los grupos de desarrollos para portar el proyecto actual a la tecnología alternativa.</li> </ul>			
<b>Actual estado:</b> 24/Enero/2015 El equipo de desarrolladores han utilizado hasta el final la biblioteca Qt, con excelentes resultados.			
<b>Desarrollado:</b> Adan Toscano López	<b>Asignado a:</b> <ul style="list-style-type: none"> <li>Andrés Sánchez Anillo</li> <li>Felipe Sánchez Arena</li> </ul>		

Documentación de información de riesgos			
ID del riesgo:	RS-04	Fecha:	10/12/2014
Prob:	60%	Impacto:	Medio
<b>Descripción:</b> Cambio de organización. Por orden superior, el grupo de proyecto SORGES sufre algún cambio en el personal de organización del proyecto.			
<b>Contexto:</b> <ul style="list-style-type: none"> <li>Contexto 1: Por orden de Dr. Carlos Rioja del Rio, jefe de los grupos de desarrollo, el grupo sufre la incorporación de un nuevo personal dedicado a la organización del proyecto.</li> </ul>			
<b>Reducción:</b> <ul style="list-style-type: none"> <li>Realizar con la mayor antelación el reparto de tareas a cada miembros del equipo, y documentar todo.</li> </ul>			
<b>Plan de contingencia:</b> <ul style="list-style-type: none"> <li>Presentar informes de la anterior organización al nuevo personal, para preservar la mayor parte de la organización anterior.</li> </ul>			
<b>Actual estado:</b> 24/Enero/2015 El equipo finalmente no ha sufrido ningún cambio.			
<b>Desarrollado:</b> Adan Toscano López	<b>Asignado a:</b> <ul style="list-style-type: none"> <li>Andrés Sánchez Anillo</li> <li>Rubén Moreno Leyva</li> <li>Felipe Sánchez Arena</li> <li>Adán Toscano López</li> </ul>		

Documentación de información de riesgos			
ID del riesgo:	RS-05	Fecha:	10/12/2014
Prob:	30%	Impacto:	Medio
<b>Descripción:</b> Perdida de algún miembro del equipo. Por motivos personales u orden superiores, el equipo pierde algún miembro del equipo.			
<b>Contexto:</b> <ul style="list-style-type: none"> <li>Contexto 1: Por orden de Dr. Carlos Rioja del Río, jefe de los grupos de desarrollo, el grupo sufre la baja de un miembro del equipo.</li> <li>Contexto 2: Por motivos personales, algún miembro del equipo deja el equipo del proyecto SORGES.</li> </ul>			
<b>Reducción:</b> <ul style="list-style-type: none"> <li>Realizar pequeños informes o exposiciones al resto de los componentes explicando los módulos desarrollados, así todos los componentes del equipo tendrán una idea general de los diferentes módulos que se desarrollan</li> </ul>			
<b>Plan de contingencia:</b> <ul style="list-style-type: none"> <li>Reorganizar los grupos de desarrollos y proporcionar la información necesaria a los nuevos desarrolladores cubrir dicha baja.</li> </ul>			
<b>Actual estado:</b> 24/Enero/2015 El equipo finalmente no ha sufrido ninguna baja.			
<b>Desarrollado:</b> Adan Toscano López	<b>Asignado a:</b> <ul style="list-style-type: none"> <li>Andrés Sánchez Anillo</li> <li>Rubén Moreno Leyva</li> <li>Felipe Sánchez Arena</li> <li>Adán Toscano López</li> </ul>		

### 3. Análisis de requisitos

#### 3.1. Especificación de los requisitos funcionales

##### 3.1.1. Requisito principal del sistema:

**RQS-001 Representación gráfica de eventos sísmicos:** el sistema ha de representar gráficamente toda la información referente a los eventos sísmicos que van aconteciendo en tiempo real o simulando eventos pasados, en un mapa geográfico entre las coordenadas siguientes:

- *Latitud desde los 34° a los 40° Norte.*
- *Longitud desde 3° a los 14° Oeste.*

##### 3.1.2. Requisitos funcionales derivados del requisito principal:

- **RQSF-001.1 Representación en tiempo real:** el sistema debe mostrar los eventos en el mapa en tiempo real y de manera dinámica para dotar al cliente de tiempo de reacción. Para mostrar los datos en tiempo real se debe ejecutar el sistema con esa opción y representará los datos que vayan apareciendo en los ficheros de entrada desde el momento en el que se inicia el sistema, y no antes.
- **RQSF-001.2 Representación en modo simulación:** el sistema debe permitir la opción de su ejecución en modo simulación, en el cual el usuario introduce los datos del evento del que quiere ver representado su proceso y el sistema debe hacer la animación como si de tiempo real se tratase.
- **RQSF-002 Representación en el mapa geográfico de las estaciones pertenecientes a la red ALERT-ES:** el sistema ha de representar en el mapa, según sus coordenadas, las estaciones de las que el módulo ALERT-ES ha enviado la información. La forma de representarlo será mediante un triángulo en la coordenada correspondiente. Este triángulo recibirá un color atendiendo a nuestra escala de magnitudes de evento, llamada *Alerte On-Site*:
  - Negro: color inicial.
  - Nivel 0 : Verde.
  - Nivel 1: Amarillo.
  - Nivel 2: Naranja.
  - Nivel 3: Rojo.

- **RQSF-003 Representación en el mapa geográfico de los orígenes del evento:** los orígenes del evento, así como el evento final, se representarán con un círculo en el epicentro del mismo.
- **RQSF-004 Cambios de color de alerta:** se cambiarán de color las estaciones a las que afecta ese origen según la información que llegará como entrada al sistema.
- **RQSF-005 Representación de la expansión de la onda:** Conforme la onda S del origen se va propagando, se debe representar esta propagación mediante una animación de círculos concéntricos a partir de epicentro del evento. Cada segundo se debe pintar un nuevo círculo de propagación, aumentando el radio en función de:  $(Tiempo\_origen - Tiempo\_actual) * Velocidad\_propagación$ . Cada vez que un nuevo origen llegue al sistema se debe representar inmediatamente, aunque ya haya una animación en curso se debe sobreescribir.
- **RQSF-005.1 Alcance de la onda.** La onda expansiva debe mostrar un degradado de colores que represente el impacto en las zonas a la que está llegando, entendido el impacto como posibilidad de afectación pero sin evaluar intensidad del seísmo. La escala para el degradado, inversa a la alerta *on-site*, sería:
  - Radio entre 0 y 30 km: color rojo.
  - Radio entre 30 y 60 km: color naranja.
  - Radio entre 60 y 120 km: color amarillo.
  - Radio mayor de 120 km: color verde.
- **RQSF-006 Finalización de la animación:** La animación también acabará o cuando los círculos topen con los límites de la área geográfica monitorizada o bien cuando pase un tiempo determinado, tiempo definido en 5 minutos desde la llegada del evento a la animación.

### 3.2. Especificación de los requisitos no funcionales

- **RQSNF-001 Requisitos de idioma:** el software debe de estar implementado en inglés, puesto que se ha considerado que hacerlo así es lo más adecuado, al desarrollarse un sistema científico y que además va a interactuar con otros sistemas también en ese idioma.
- **RQSNF-002 Requisitos de disponibilidad:** el software siempre debe estar en funcionamiento pues los eventos pueden ocurrir a cualquier hora.

- **RQSNF-003 Requisitos de fiabilidad:** el sistema debe aportar máxima fiabilidad en la representación de los eventos sísmicos, pues al ser un software científico implica alta precisión.
- **RQSNF-004 Requisitos de implementación:** el software se implementará en el lenguaje C++ por compatibilidad con el sistema ya instalado.
- **RQSNF-005 Requisitos de seguridad:** se realizará un backup que almacenaremos en forma de log con todos los eventos acontecidos incluyendo sus orígenes y estaciones.

### **3.3. Especificación de los requisitos de rendimiento**

El rendimiento del sistema debe asegurar la correcta captación del flujo de datos sin que el sistema se resiente o quede inoperativo y la representación ininterrumpida en el mapa. El rendimiento ocupa un papel clave pues con su buena optimización dotará al cliente de tiempo para operar con ventaja en función de lo que se esté representando.

### **3.4. Actores del proyecto**

Se considera que los actores del proyecto son las personas que van a interactuar con el software, por lo tanto establecemos como actores del proyecto a las siguientes personas:

Por parte del Real Observatorio de la Armada de San Fernando:

- El Comandante D.Antonio Pazos.
- Los técnicos informáticos del Observatorio.

Además, los miembros de nuestro equipo de desarrollo también interactuarán con el producto final en las primeras fases de su implantación al menos. Los integrantes se pueden consultar en el apartado 2.3 *Organización*.

### **3.5. Estudio de tecnologías alternativas**

El cliente ya tiene instalado en su sistema el software **seiscomp3** que se trata de un software sísmico completo que incluye la adquisición de la forma de la onda sísmica, detección automática de terremoto, ubicación del origen y su cuantificación, reubicación de evento manual, evento de alerta; y el archivo de forma de onda y difusión. Permite representar los eventos en el mapa tal y como buscamos nosotros, pero estos eventos los representa una vez ya han pasado en el tiempo, no en tiempo real, y no se representa el proceso de orígenes ni cambios de color de las estaciones.

En el caso de ALERT-ES, no es un módulo con salida gráfica, por lo que no puede representar en mapa sus datos.

### **3.6. Análisis GAP**

Si analizamos las deficiencias específicas de SEISCOMP3 y ALERT-ES, encontramos varios puntos en los que nuestro software da un paso al frente:

1. La necesidad de representación en tiempo real surge tras el desarrollo del módulo ALERT-ES, ya que se busca que se puedan representar los datos de este sistema de alerta temprana. Y es esta necesidad la que busca cubrir nuestro software. Esta nueva adición se traducirá en un salto cualitativo del sistema, ya que ALERT-ES no dispone de una salida gráfica para ver en un mapa los eventos que recibe.
2. SEISCOMP3 puede llegar a representar en tiempo real los eventos, pero únicamente los eventos, es decir, la última parte de la secuencia. No representa los orígenes intermedios, cosa que nuestro software sí hace.
3. Además, al no representar el proceso completo, SEISCOMP3 no cambia el color a las estaciones dependiendo de los picks, cosa que nuestro software sí.
4. La expansión de la onda mediante círculos concéntricos de los orígenes/eventos también es algo de lo que carecen los otros sistemas, y más aún de la representación del alcance por colores dependiendo de la distancia radial.

## **4. Diseño**

### **4.1. Diseño conceptual**

Este modelo podría ir incluido también en el apartado de análisis, ya que en base es un paso a medio camino entre el análisis, de los diferentes tipos de requisitos y el dominio del sistema, y el diseño del propio sistema.

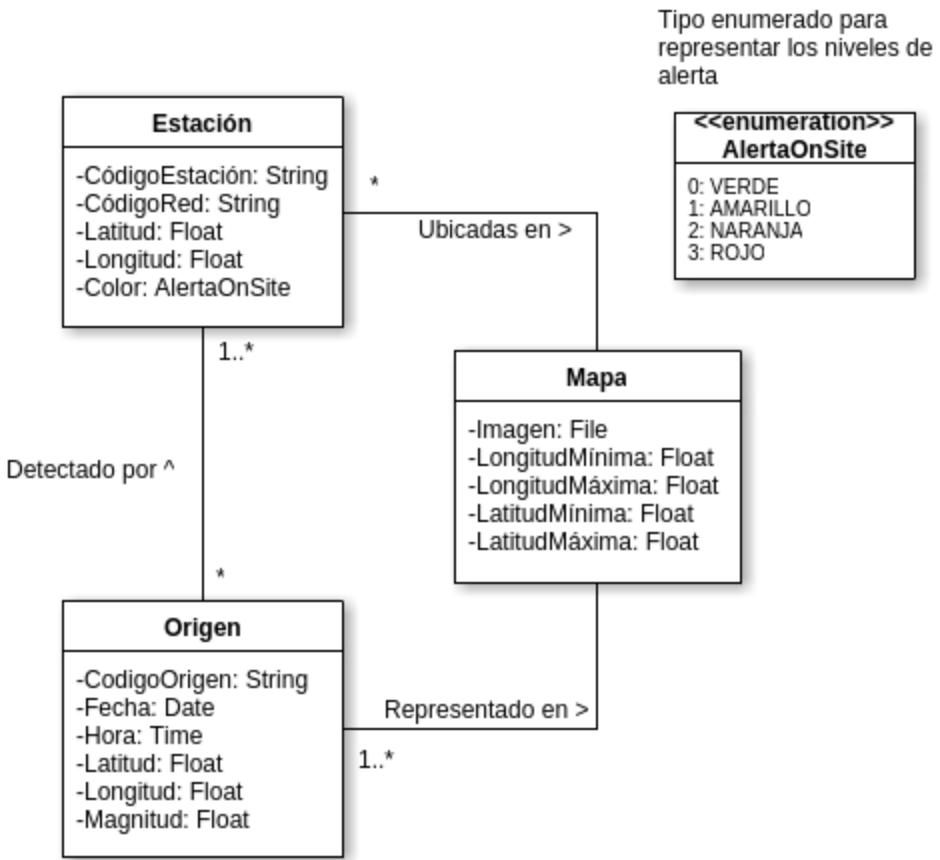
En este diagrama, realizando siguiendo la notación UML, se detalla la interpretación que se le ha dado a las entidades de datos tratadas en el dominio del problema y sus relaciones.

El sistema va a trabajar con datos relativos a 3 entidades base: los orígenes sísmicos, las estaciones receptoras y el mapa de la zona geográfica. Cada una de ellas tiene sus atributos respectivos reflejados en el modelo.

En cuanto a las relaciones entre ellas:

1. Relación Origen-Estación.
  - Cada Origen detectado lo será porque al menos 1 Estación habrá recibido señales que así lo certifiquen (cardinalidad 1..\*).
  - Una misma estación puede participar en la detección de varios orígenes, por definición no simultáneos en el tiempo, o bien puede ser que nunca reciba señales sísmicas de origen alguno (cardinalidad \*).
2. Relación Mapa-Estación.
  - Este sistemas por definición sólo trabajará sobre un mapa concreto, por lo que será este mapa en el que las estaciones deben estar ubicadas (cardinalidad 1 omitida).
  - El mapa, inicialmente puede no contener estaciones (cardinalidad \*).
3. Relación Mapa-Origen.
  - Este sistemas por definición sólo trabajará sobre un mapa concreto, por lo que será este mapa en el que el origen debe ser representado (cardinalidad 1 omitida).
  - El mapa, inicialmente puede no estar representando ningún origen y como máximo representará solo 1 Origen simultáneamente (cardinalidad 1.\*).

A tener en cuenta también un tipo de datos específico definido para representar los niveles de alerta *On Site* y su correspondiente escala de colores, que van a ser los que marquen la intensidad de las ondas sísmicas que llegan a cada estación. (Tipo enumeration: *AlertaOnSite*).



## 4.2. Diseño lógico

Se ha decidido seguir una arquitectura para el sistema dividida por módulos según la funcionalidad requerida, y que da como resultado los 2 siguientes:

- **El módulo de procesamiento de datos:** Este módulo se encargará de procesar los datos de estaciones, picks, orígenes y eventos que se generarán en los ficheros de los sistemas SEISCOMP3 y ALERTES. Estos ficheros serán: por un lado, los logs de actividad de ALERTES en los que se registrarán las llegadas de picks a las estaciones y de orígenes detectados por estas; y por el otro, los ficheros en formato xml que SEISCOMP3 genera cuando se procesan los eventos.  
Una vez se defina la ruta de acceso para el control de los ficheros de datos, será este módulo el que conectará con el sistema de ficheros y extraerá de cada uno la información necesaria para crear las estructuras de datos con las que trabajará la aplicación: creará la lista de estructuras representativas de las estaciones y hará lo mismo cuando se reciba un origen con su estructura de datos.

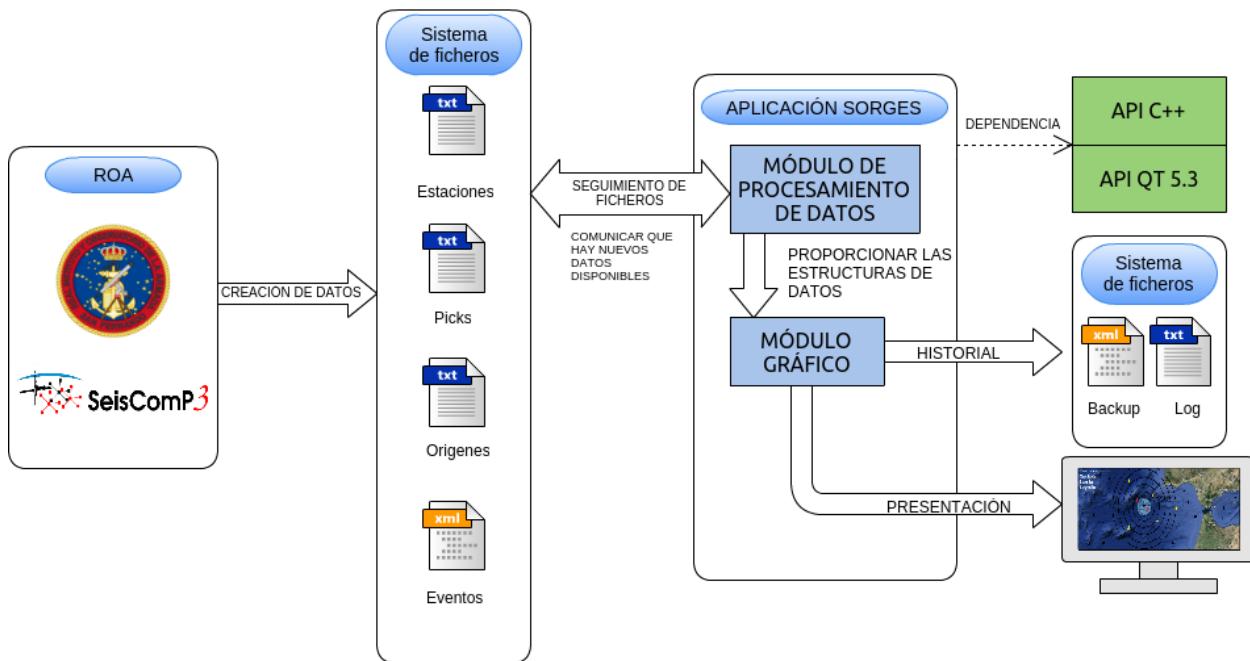
El software trabajará únicamente con estructuras de datos almacenadas en la memoria principal del sistema, por lo que no será necesaria una interacción con base de datos alguna.

Una vez este módulo ha procesado los datos y construido las estructuras, será el encargado de transmitirle estas al módulo gráfico.

- **El módulo gráfico:** tras recibir las estructuras de datos del módulo de procesamiento, este módulo gráfico será el encargado de presentar estos datos a través de la interfaz de usuario pintando las diferentes estaciones y orígenes en el mapa y mostrando la información de los atributos.  
La aplicación podrá funcionar en dos modos distintos: ***tiempo real***, en el cuál procesará la información que vaya llegando a los ficheros de datos desde el inicio del sistema; o ***simulación***, en el cuál se mostrarán los datos de un evento concreto indicado por el usuario de entre todos los ya registrados por ALERTES y SEISCOMP3.

En el siguiente modelo se representa el funcionamiento lógico de la aplicación.

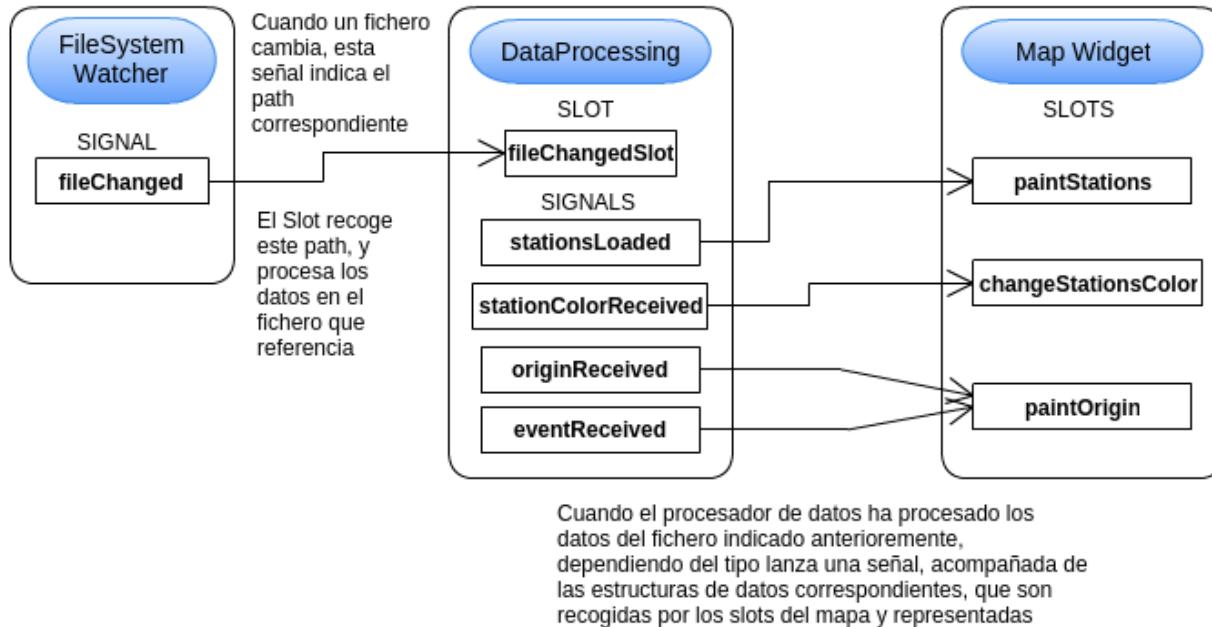
El funcionamiento en cada uno de los modos de ejecución es el mismo, con la única diferencia de que en *tiempo real* la aplicación espera hasta que aparecen nuevos datos para procesarlos y cuando la ejecución es en *simulación* es la aplicación misma la que rastrea los datos de la fecha pedida por el usuario en los ficheros y los procesa.



Para manejar el funcionamiento de la aplicación en sus dos modos de ejecución, se ha diseñado un sistema de señales, aprovechando la funcionalidad de SIGNALS y SLOTS que proporciona la biblioteca Qt.

Con este sistema, se pueden definir señales (*signals*) que se emitirán cuando sucedan ciertos eventos o condiciones, y que activarán al enviarse ciertas funciones (*slots*) al igual que si estos métodos fueran llamados directamente por un objeto.

A continuación se muestra un diagrama ilustrativo del funcionamiento de este sistema en relación al proceso desde que los datos se generan en el fichero, se procesan y se representan posteriormente.

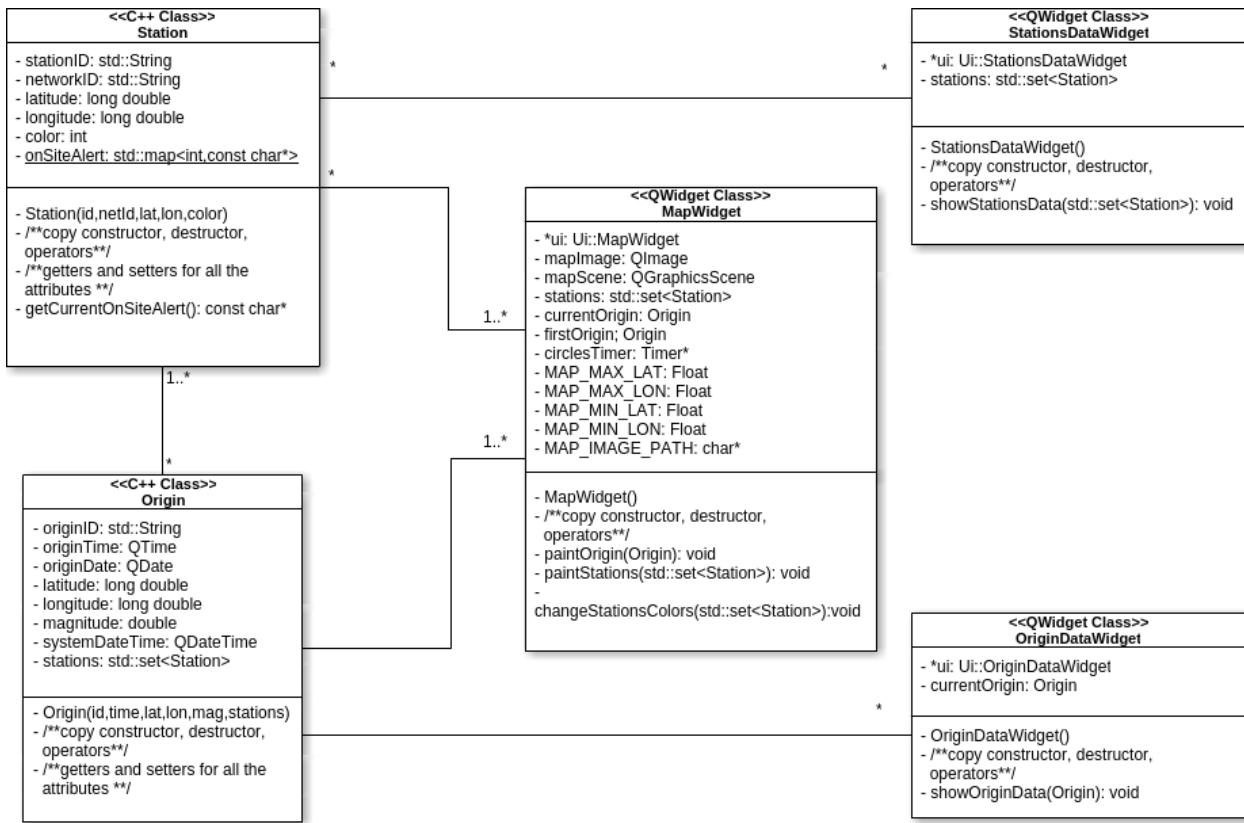


### 4.3. Diseño físico

En este apartado se define el modelo de clases que van a ser usadas como estructura de datos para la aplicación. Al ser este el diseño más cercano a la implementación, se usará en los diagramas la nomenclatura y los tipos proporcionados por el lenguaje C++ en general y por el API específico de Qt 5.3 .

Para facilitar la lectura, se ha decidido separar el diagrama de clases de diseño en 2 subdiagramas siguiendo la división por módulos de la aplicación. Las clases que aparecen en cada subdiagrama son las que intervienen en el módulo específico, pudiendo estar presentes en varios de ellos.

### 4.3.1. Módulo gráfico.



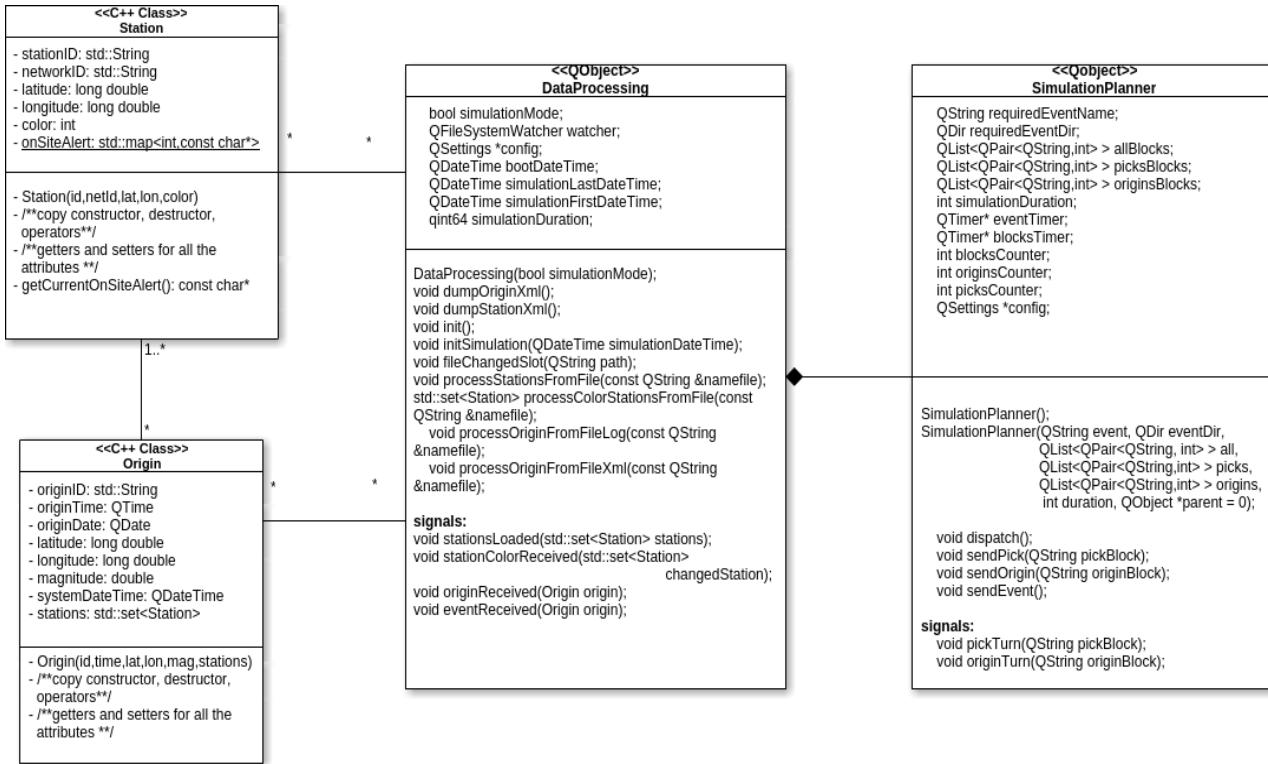
A continuación se detallarán los atributos y métodos públicos presentes en cada clase. En cuanto a la explicación de los métodos presentes en las clases del diagrama, se obvian los constructores, destructores, operadores y *getters and setters* por su funcionamiento estándar.

- **Clase Station:**
  - *stationID*: atributo de tipo string que corresponde al código de la estación.
  - *networkID*: atributo de tipo string que corresponde al código de la red.
  - *latitude*: atributo de tipo long double que representa la latitud en la que se encuentra la estación (grados en decimal).
  - *longitude*: atributo de tipo long double que representa la longitud en la que se encuentra la estación (grados en decimal).
  - *color*: atributo de tipo entero que representa el color en el código de alerta de colores de la estación.
  - *onSiteAlert*: atributo estático (de clase) de tipo map que hace corresponder cada entero (primer miembro) con su cadena de caracteres (valor hexadecimal del color) correspondiente color en la escala de alerta (segundo miembro).

- `getCurrentOnSiteAlert()`: devuelve el color (tipo `const char *`) almacenado en el atributo de clase `onSiteAlert` correspondiente al código de alerta reflejado en el atributo `color`.
- **Clase *Origin*:**
  - `originID`: atributo de tipo `string` que corresponde al código del origen.
  - `originTime`: atributo de tipo `QTime`, estructura que almacenará la hora en la que se ha producido el origen.
  - `originDate`: atributo de tipo `QDate`, estructura que almacenará la hora en la que se ha producido el origen.
  - `systemDateTime`: atributo de tipo `QDateTime`, estructura que almacenará la marca de tiempo (fecha y hora) en la que el origen ha sido procesado por el sistema.
  - `latitude`: atributo de tipo `long double` que representa la latitud en la que se encuentra el epicentro del origen (grados en decimal).
  - `longitude`: atributo de tipo `long double` que representa la longitud en la que se encuentra el epicentro del origen (grados en decimal).
  - `magnitude`: atributo de tipo `double` que representa la magnitud en la escala Richter del origen.
  - `stations`: atributo de tipo `set` (conjunto) que almacenará los objetos de tipo `Station` correspondientes a las estaciones que han detectado el origen.
- **Clase *MapWidget*:**
  - `ui`: atributo de tipo `Ui::MapWidget *`, que no es otra cosa que la estructura que QtGUI proporciona para la interfaz de usuario.
  - `mapImage`: atributo de tipo `QImage` que representa la imagen del mapa.
  - `mapScene`: atributo de tipo `QGraphicsScene`, que no es más que la representación de la escena o zona de la interfaz gráfica donde se va a mostrar la imagen del mapa.
  - `stations`: atributo de tipo `set` (conjunto) que almacenará los objetos de tipo `Station` correspondientes a las estaciones presentes en el mapa.
  - `currentOrigin`: atributo de tipo `Origin` que corresponderá al origen actualmente representado en el mapa (en el caso de no estar representando ninguno, el atributo será nulo).
  - `firstOrigin`: atributo de tipo `Origin` que corresponderá al primer origen de la secuencia de un evento que es representado en el mapa (en el caso de no haberse iniciado secuencia alguna o de haber terminado esta, será nulo).
  - `circlesTimer`: atributo de tipo `QTimer`, un temporizador que controlará la generación de círculos concéntricos representativos de la expansión de la honda.
  - `MAP_MAX_LAT`: atributo de tipo `float` que representa los grados decimales correspondientes a la máxima latitud del mapa (máximo valor eje Y).

- MAP\_MAX\_LON: atributo de tipo float que representa los grados decimales correspondientes a la máxima longitud del mapa (máximo valor eje X).
  - MAP\_MIN\_LAT: atributo de tipo float que representa los grados decimales correspondientes a la mínima latitud del mapa (mínimo valor eje Y).
  - MAP\_MIN\_LON: atributo de tipo float que representa los grados decimales correspondientes a la mínima longitud del mapa (mínimo valor eje X).
  - MAP\_IMAGE\_PATH: cadena de caracteres con la ruta correspondiente al archivo de imagen con el mapa.
  - paintOrigin(Origin): método encargado de representar el origen.
  - paintStations(std::set<Station>): método encargado de pintar las estaciones en el mapa.
  - changeStationsColor(std::set<Station>): método encargado de cambiar el color a las estaciones (o estación) indicada(s).
- Clase **OriginDataWidget**:
    - *ui*: atributo de tipo *Ui::OriginDataWidget* \*, que no es otra cosa que la estructura que QtGUI proporciona para la interfaz de usuario.
    - *currentOrigin*: atributo de tipo Origin que corresponderá al origen actualmente representado en el mapa (en el caso de no estar representando ninguno, el atributo será nulo).
    - showOriginData(Origin): método encargado de presentar en la interfaz los datos del origen.
  - Clase **StationsDataWidget**:
    - *ui*: atributo de tipo *Ui::StationsDataWidget* \*, que no es otra cosa que la estructura que QtGUI proporciona para la interfaz de usuario.
    - *stations*: atributo de tipo set (conjunto) que almacenará los objetos de tipo Station correspondientes a las estaciones presentes en el mapa.
    - showStationsData(std::set<Station>): método encargado de presentar en la interfaz los datos de las estaciones.

#### 4.3.2. Módulo de procesamiento de datos.



En este módulo aparecen 2 clases nuevas: *DataProcessing*, que va a ser la encargada de interpretar y procesar los datos que el módulo de conexión almacenó provenientes de ALERT-ES y con ellos crear las estructuras de datos, que serán los objetos de las clases *Origin* y *Station*; de ahí que estas dos clases aparezcan en el diagrama con líneas de dependencia, ya que, aunque no tienen una asociación directa con *DataProcessing*, sí que esta clase depende del diseño de ellas ya que tiene que crear los objetos siguiendo la estructura definida.

Además, está *SimulationPlanner*, una clase que se relaciona mediante composición con *DataProcessing* y que será la encargada de planificar y ejecutar la generación simulada de datos que la clase procesadora le indicará durante el modo de simulación de la aplicación.

A continuación se detallarán los atributos y métodos públicos presentes en estas clases. En cuanto a la explicación de los métodos presentes en las clases del diagrama, se obvian los constructores, destructores, operadores y *getters and setters* por su funcionamiento estándar.

- Clase **DataProcessing**:

- *init()*; inicializador del procesador.
- *initSimulation(QDateTime)*: inicializador del procesador para la simulación de los datos correspondientes a la fecha indicada.
- *fileChangedSlot(QString path)*:
- *processStationsFromFile(const QString &file)*: procesará los datos de las estaciones disponibles desde el fichero que se recibirá cuando se inicie la aplicación y creará de ellos un objeto *std::set<Station>* que contendrá todos los objetos *Station* y que será usado por el módulo gráfico posteriormente.
- *processColorStationsFromFile(const QString &file)*: procesará el fichero de log de picks para cambiar los colores de alerta de las estaciones cuando estos lo indiquen.
- *processOriginFromFileLog(const QString &file)*: procesará los datos del fichero de log de orígenes y creará de ellos objetos *Origin* que serán usados por el módulo gráfico posteriormente.
- *processOriginFromFileXml(const QString &file)*: procesará los datos de los ficheros xml de eventos y creará de ellos objetos *Origin* representativos de estos eventos que serán usados por el módulo gráfico posteriormente.
- *simulationMode*: indicador booleano de si el procesador está trabajando en modo simulación o por el contrario en tiempo real.
- *origin*: último objeto origen procesado.
- *stations*: último conjunto de estaciones procesadas
- *watcher*: rastreador del sistema de ficheros local que vigilará los cambios producidos en los logs y xml de los cuales hay que procesar los datos.
- *config*: objeto que referencia el archivo de configuración de la aplicación.
- *bootDateTime*: tiempo de inicio del procesador.
- *simulationLastDateTime*: en modo simulación, la última fecha del rango para procesar los datos en los registros.
- *simulationFirstDateTime*: en modo simulación, la primera fecha del rango para procesar los datos en los registros.
- *simulationDuration*: periodo total del rango de la simulación en milisegundos.

- Clase **SimulationPlanner**:

- *dispatch()*: método que controlará los temporizadores para enviar y volcar los datos de picks u orígenes a los ficheros controlados por el módulo gráfico para su representación, manteniendo la secuencia de tiempos en la cuál estos datos fueron creados, creando una simulación de tiempo real así.
- *sendPick(QString pickBlock)*: método que envía los datos de un pick concreto.
- *sendOrigin(QString originBlock)*: método que envía los datos de un origen concreto.

- *sendEvent()*: método que envía los datos del evento final de la simulación.
- *requiredEventName*: nombre del evento para el que se está realizando la simulación.
- *requiredEventDir*: directorio donde se encuentra el fichero xml del evento.
- *allBlocks*: bloques de datos tanto de orígenes como picks procesados desde los logs por el procesador, siguiendo una estructura por la cual se almacenan mediante una lista de pares la cadena con los datos y un entero que especifica la diferencia en milisegundos de su ocurrencia respecto al bloque de datos que lo precede.
- *picksBlocks*: bloques de datos de picks procesados desde los logs por el procesador, siguiendo una estructura por la cual se almacenan mediante una lista de pares la cadena con los datos y un entero que especifica la diferencia en milisegundos de su ocurrencia respecto al bloque de datos que lo precede.
- *originsBlocks*: bloques de datos de orígenes procesados desde los logs por el procesador, siguiendo una estructura por la cual se almacenan mediante una lista de pares la cadena con los datos y un entero que especifica la diferencia en milisegundos de su ocurrencia respecto al bloque de datos que lo precede.
- *simulationDuration*: periodo total del rango de la simulación en milisegundos
- *blocksTimer*: temporizador que modela las diferencias de tiempos para el envío de datos entre bloques.
- *blocksCounter*: contador que controla el número de bloques de datos que se van enviando.
- *originsCounter*: contador que controla el número de bloques de datos de orígenes que se van enviando.
- *picksCounter*: contador que controla el número de bloques de datos de picks que se van enviando.
- *config*: objeto que referencia el archivo de configuración de la aplicación.

#### 4.3.3. Diseño de la interfaz de usuario.

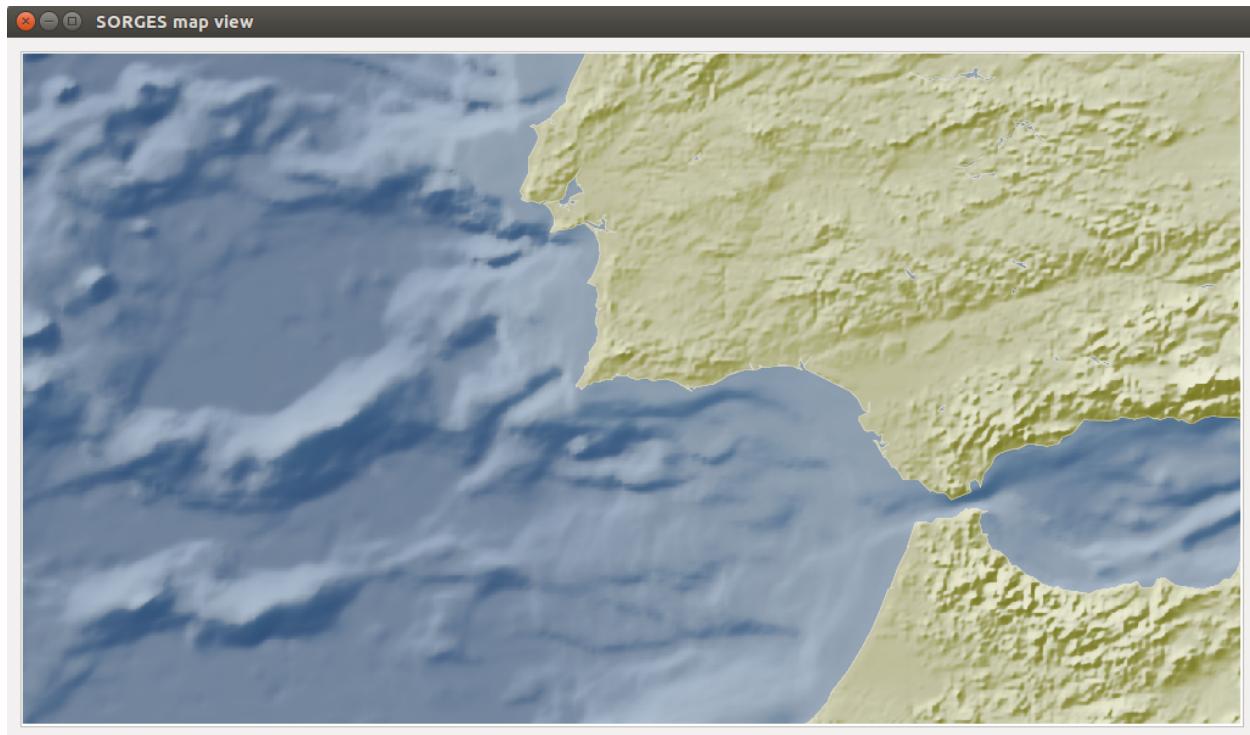
Para diseñar la interfaz de usuario se ha utilizado la herramienta *QtDesigner* a través del IDE *QtCreator*.

La base principal de la interfaz son 3 *widgets*, que son las clases proporcionadas por QtGUI para las sub-ventanas incluidas dentro de la ventana principal o bien ventanas independientes.

En un principio se intentó diseñar la aplicación sobre una ventana principal con todos los datos, pero debido al tamaño del mapa se ha decidido por claridad disponer estos 3 widgets de forma independiente como ventanas individuales.

Estos 3 widgets están reflejados en el diagrama de clases de diseño: un widget para mostrar los datos de las estaciones presentes en el mapa, otro para mostrar la información del origen que se está representando y un último widget que contiene la vista del mapa.

Las interfaces quedarían de la siguiente manera:



**SORGES origin data**

Origin ID

Timestamp

Epicenter: latitude   
longitude

Magnitude

Related Stations

**SORGES list of stations**

List of current stations

## 5. Implementación

### 5.1. Entorno tecnológico

#### Estructura del IDE

Con el fin de permitir el análisis del proyecto, su posible ampliación o mejora, por parte del personal técnico del Real Observatorio de la Armada de San Fernando, se explicará la estructura que tiene, para así facilitar su entendimiento.

En primer lugar, se hablará sobre la biblioteca utilizada en el proyecto en casi su totalidad, llamada QT, y las bibliotecas utilizadas, así como su aportación funcional a nuestras funcionalidad del proyecto. Qt es una biblioteca multiplataforma, distribuida bajo Lesser General Public License (LGPL) y que cuenta con una gran cantidad de documentación. Fue desarrollada por la división de software Qt de Nokia, y está diseñada para usarse sobre el lenguaje C++, aunque ofrece herramientas para utilizarse sobre otros lenguajes. Qt también proporciona un entorno integrado de desarrollo (IDE) llamado QtCreator, que además tiene integrado el editor de interfaces gráficas QtDesigner.



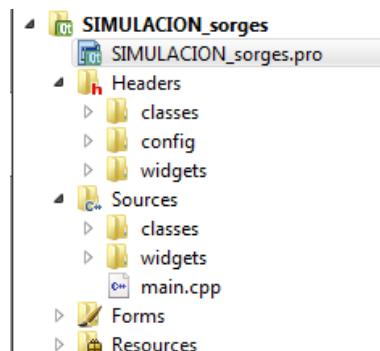
Bibliotecas de Qt utilizadas para el proyecto:

- **QTime:** Estructura de datos utilizadas para almacenar horas, minutos, segundos y milisegundos con 3 dígitos de precisión.
- **QDate:** Estructura de datos utilizadas para almacenar dia, mes y año correspondiente a una fecha.
- **QDateTime:** Estructura de datos utilizadas para almacenar dia, mes, año, hora, minutos, segundo y milisegundos con 3 dígitos de precisión.
- **QObject:** Objeto base de todos los objetos de la biblioteca. Usado para implementar las funcionalidades básicas como el sistema de señales.

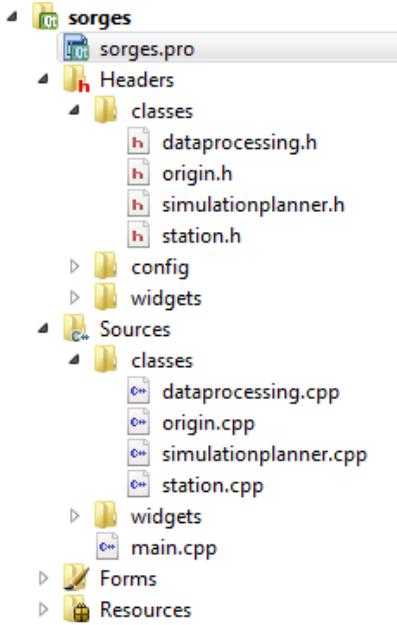
- QRegExp: Estructura de datos que nos permite realizar búsqueda en un QString mediante expresiones regulares, principal motor de búsqueda en nuestro módulo de procesamiento de datos.
- QFile: Estructura de datos que nos permite abrir un fichero, para leer o escribir en él.
- QDir: Estructura de datos para la gestión de directorio en el sistema huésped.
- QStringList: Estructura de datos de lista de QString, fácil de insertar y recorrer todos sus elementos.
- QString: Estructura de datos equivalente al típico std::String, pero con algunas ventajas aportadas por los desarrolladores de Qt, dentro de las ventajas, la búsqueda de expresiones regulares mediante esta estructura, es la que más ha aportado al proyecto.
- QFileSystemWatcher: Conjunto de funciones, esta mediante threads y demonios controlan cualquier cambio de un fichero o directorio (ya sea modificado, creado o borrado), muy útil para determinar cambios en los ficheros del sistema y procesarlos a tiempo real.
- QSettings: Estructura de datos que nos permite guardar variables en ficheros externos al programa de forma persistente, utilizado en el proyecto para almacenar los Path de los ficheros que debemos procesar a tiempo real, y los ficheros temporales generados por la simulación.
- QDomDocument/QXml: Estructura de datos que nos permite leer un fichero anteriormente abierto mediante la biblioteca QFile, muy útil para procesar el fichero del sistema proporcionado en formato XML.
- QTextStream: Estructura de flujos de datos, concretamente de fujo de texto, normalmente asociado a QString, utilizado para escribir datos en ficheros.
- QtGui: base de la biblioteca de elementos gráficos de Qt, usada tanto directamente como a través de QtDesigner.
- QWidget: base para implementar los widgets de la aplicación, que heredan de ella.
- QGraphicsScene: escena de gráficos para disponer los ítems como la imagen del mapa, las estaciones, los círculos, etc.

- QTimer: Estructura de datos que simula la funcionalidad de un temporizador, que nos permite ejecutar funciones de forma paralela (mediante el uso de SIGNAL y SLOT), utilizada para realizar la animación del mapa de eventos sísmicos y para generar la simulación de eventos anteriores.
- QGraphicsItem: clase base de los elementos gráficos pintados, como pueden ser los polígonos que representan las estaciones, o los círculos de epicentro u onda expansiva de los orígenes y eventos.
- QWhatsThis: funcionalidad de Qt que permite mostrar una etiqueta informativa al hacer *click* en los elementos gráficos del mapa.

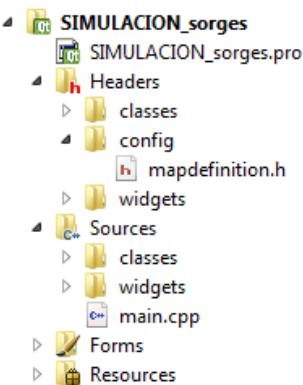
Ahora vamos a hablar de la estructura interna que tiene el proyecto, hablando de cada uno de los módulos que hemos dividido el proyecto. En primer lugar, en el proyecto nos lo encontramos dividido en varios bloques: *Headers*, *Sources*, *Forms* y *Resources*.



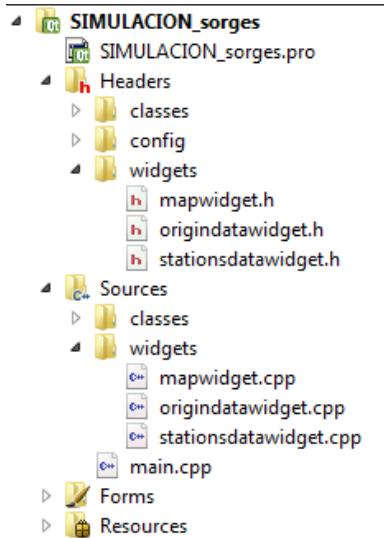
Dentro de Headers tendremos la declaración de cada una de las clases de cada módulo definido en el proyecto. En Sources nos encontramos las implementaciones de cada función declaradas anteriormente en la sección Headers. En Form situamos los ficheros correspondientes a cada ventana de la interfaz gráfica del proyecto. Y por último en este nivel, tenemos la carpeta Resources, que se encuentran todos los ficheros internos que utilizaremos en nuestra aplicación, como por ejemplo el fichero de configuración, logo, o la imagen del mapa de la pantalla principal.



El conjunto de ficheros alojados en el directorio *classes*, son todas aquellas relacionadas directamente con las clases utilizadas para el procesamiento de datos, como pueden ser *Station* y *Origin*, como principales clases en el proyecto, ya que serán estos los que en su estructura de datos almacenen los datos obtenidos a partir de ficheros de procesamiento. También tenemos la clase *DataProcessing*, para las tareas de monitorización de modificaciones del fichero, y posterior procesamiento. Por último, tenemos *SimulationPlanner*, como clase encargada de realizar la simulación de entradas de datos en el sistema, del mismo modo que lo proporciona el sistema Seiscomp3 junto a ALERT-es.



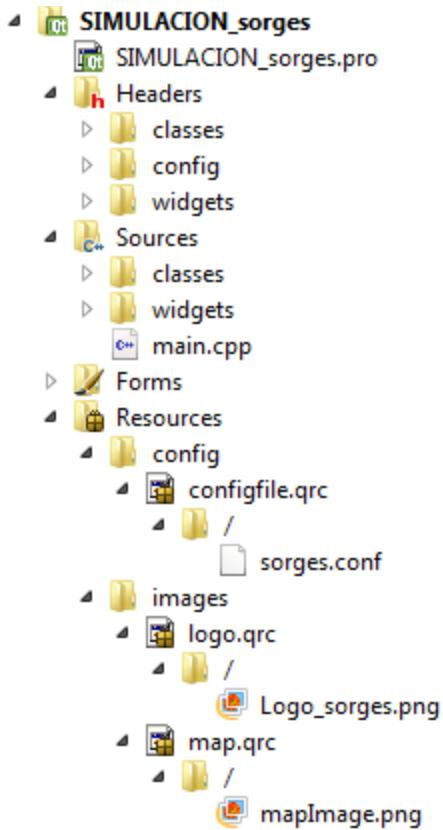
Dentro de la raíz de directorio *Config*, nos encontramos un archivo definido por el equipo de desarrollo, como fichero de constantes del sistema, donde almacenamos constantes relevantes del proyecto, desde la constante de la velocidad propagación de la onda sísmica, hasta los colores RGB empleados para la representación de las estaciones o orígenes, a grandes rasgos, todo lo definido como macros del sistema, se encuentran definidas en este fichero.



Dentro de los directorios llamados *widgets*, contienen 3 clases, *MapWidget*, encargada de la representación del mapa por pantalla y toda la información relevante sobre estaciones y orígenes; también tenemos la clase *OriginDataWidget*, encargada de la impresión de los datos por pantalla; y por último, la clase *StationsDataWidget* encargándose de la impresión por pantalla de los datos relacionados con las estaciones que recibimos.

El directorio *Form* contiene los ficheros dedicados a la configuración de pantallas gráficas del sistema, en este caso tendremos una por cada pantallas de información del proyecto, osea 3, *MapWidget*, *OriginDataWidget*, *StationDataWidget*.

Y por último, tenemos el directorio *Resources*, carpeta utilizada para el almacenamiento interno de la aplicación de recursos, como puede ser un fichero de configuración, e imágenes. En el directorio *config*, nos encontramos *sorges.conf*, como fichero de configuración del sistema, en él almacenamos algunas variables como el path de ficheros de procesamiento real, o el path de ficheros que generamos en la simulación de eventos sísmicos. Dentro de la carpeta *images/logos* nos encontramos la imagen correspondiente al logo oficial de la aplicación, y finalmente, en *images/map* encontraremos la imagen correspondiente al mapa, cabe comentar, que cualquier cambio de la imagen del mapa, conlleva la modificación de numerosas macros del fichero de configuración *mapdefinition.h* para la correcta funcionalidad de la aplicación.



## 5.2. Código fuente

Control de versiones del código.

Para controlar los cambios del código fuente del proyecto se ha usado el sistema de control de versiones *GIT*, y se ha creado un repositorio en la forja *GitHub* para alojar el proyecto.

Algoritmo de conversión de coordenadas geográficas a píxeles.

Uno de los principales focos de la implementación hasta el momento ha sido la del algoritmo destinado a realizar la conversión entre las coordenadas geográficas de latitud y longitud en los píxeles correspondientes en nuestra imagen para que puedan ser pintados en el mapa los orígenes, los círculos y las estaciones en el lugar geográfico que les corresponde.

Para realizar esta conversión, hace falta hacer una interpolación de los datos para que el sistema de referencia terrestre de coordenadas pase a ser interpretado con el sistema de coordenadas dentro de la escena gráfica proporcionada por Qt.

Esta interpolación se ha hecho de forma lineal, esto es, sin tener en cuenta la curvatura terrestre, ya que para la zona geográfica con la que se trabaja en este caso las distancias no son los suficientemente grandes como para sufrir desviaciones, algo que exigiría una interpolación que sí incluyera las variables curvilíneas.

Para implementar el algoritmo, hay que tener en cuenta el sistema de coordenadas de referencia de Qt, en el cual el píxel 0 ( $x = 0$ ,  $y = 0$ ) se encuentra en la esquina superior izquierda.

Sabiendo esto, hay que definir en términos de píxeles y coordenadas la correspondencia entre los puntos de referencia que nos servirán para la interpolación:

- La longitud se representará en el eje X y la latitud en el eje Y.
- La esquina superior izquierda será la correspondiente a las coordenadas LATITUD MÍNIMA y LONGITUD MÍNIMA, correspondiendo con el pixel (0,0).
- La esquina inferior derecha será la correspondiente a las coordenadas LATITUD MÁXIMA y LONGITUD MÁXIMA, correspondiendo con el píxel máximo tanto en el eje X como el Y.
- La fórmula para la interpolación tendría los siguientes parámetros:
  - Longitud/Latitud que se quiere convertir a píxeles: *targetLon*, *targetLat*.
  - Longitud/Latitud mínima de referencia (en el píxel de la esquina izquierda superior ): *minLon*, *minLat*.
  - Longitud/Latitud máxima de referencia (en el píxel en la esquina inferior derecha ): *maxLon*, *maxLat*.
  - Mínimo pixel del eje X/Y (que será (0,0) en este sistema de referencia): *minXpixel*, *minYpixel*.
  - Máximo pixel del eje X/Y: *maxXpixel*, *maxYpixel*.

Con esto, la fórmula quedaría de la siguiente forma (para el eje Y sólo habría que cambiar longitud por latitud):

```
pixelX = ((targetLon - minLon) / (maxLon - minLon)) * (maxXpixel - minXpixel)
```

Una vez tenemos estas restricciones, el algoritmo es el siguiente:

```
void MapWidget::coordinatesToPixels(long double &pixelX, long double &pixelY,  
                                     long double targetLat,  
                                     long double targetLon)  
{  
    long double minLon = MAP_MIN_LONGITUDE;  
    long double minLat = MAP_MIN_LATITUDE;  
    long double maxLon = MAP_MAX_LONGITUDE;  
    long double maxLat = MAP_MAX_LATITUDE;  
    long double maxXpixel = mapScene.width();  
    long double maxYpixel = mapScene.height();
```

```
long double minXpixel = 0;
long double minYpixel = 0;

pixelX = ((targetLon - minLon) / (maxLon - minLon)) * (maxXpixel -
minXpixel);
pixelY = ((targetLat - minLat) / (maxLat - minLat)) * (maxYpixel -
minYpixel);
}
```

## 6. Pruebas

### 6.1. Pruebas unitarias

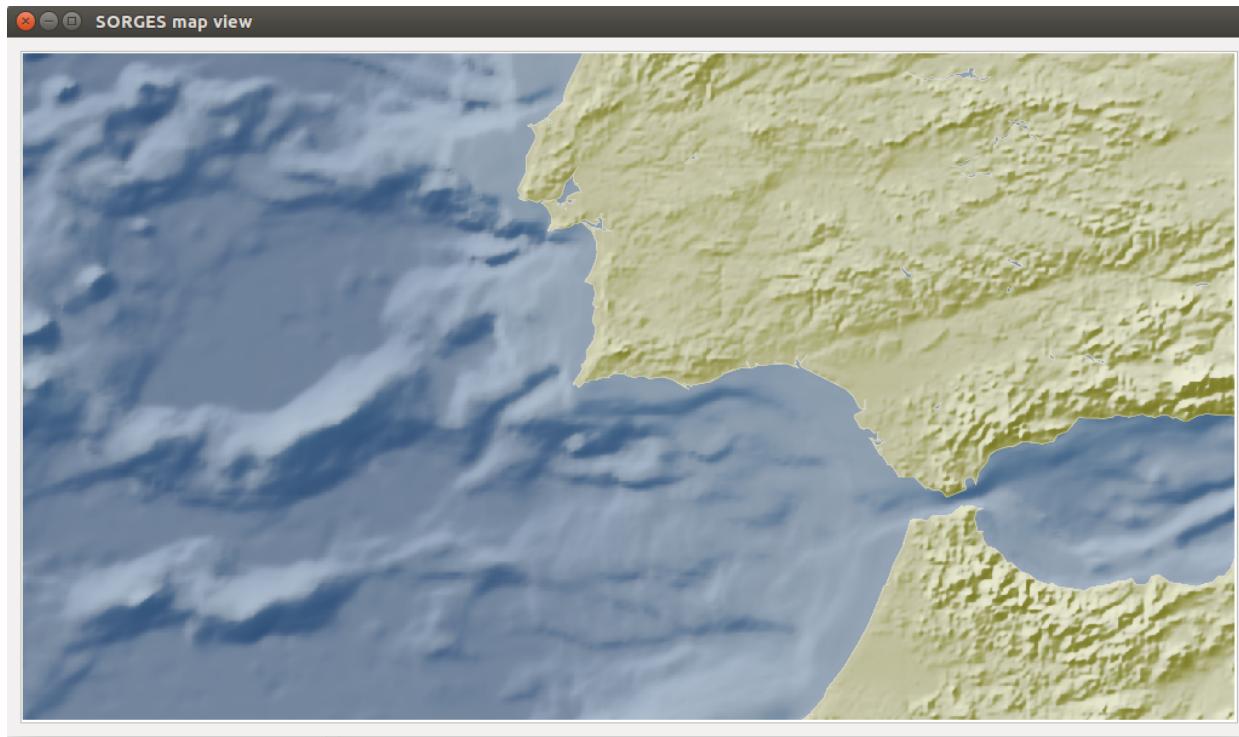
#### 6.1.1. Módulo gráfico.

- Pruebas de widgets.

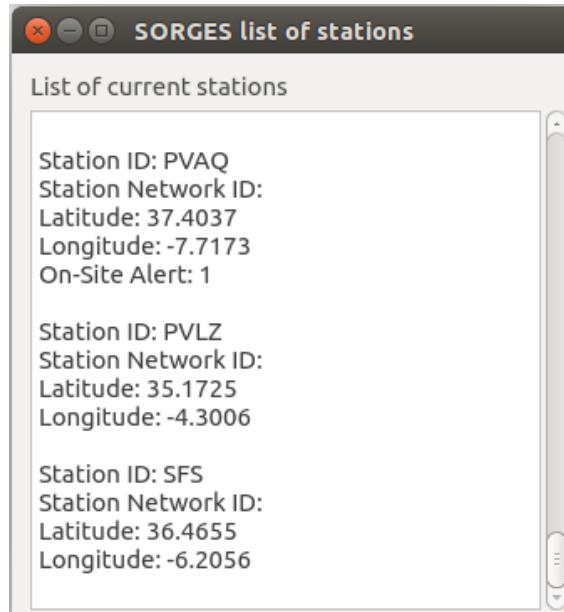
##### Pruebas de visualización de imagen.

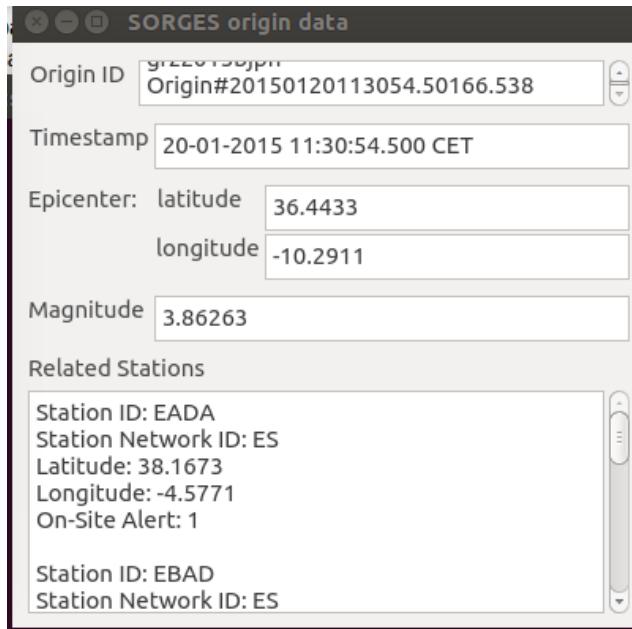
Una vez implementada la clase MapWidget y sus atributos y métodos, se ha procedido a probar la correcta visualización de la imagen seleccionada para representar el mapa y su correcto ajuste en tamaño al widget.

El resultado ha sido satisfactorio como se puede ver en la siguiente captura.



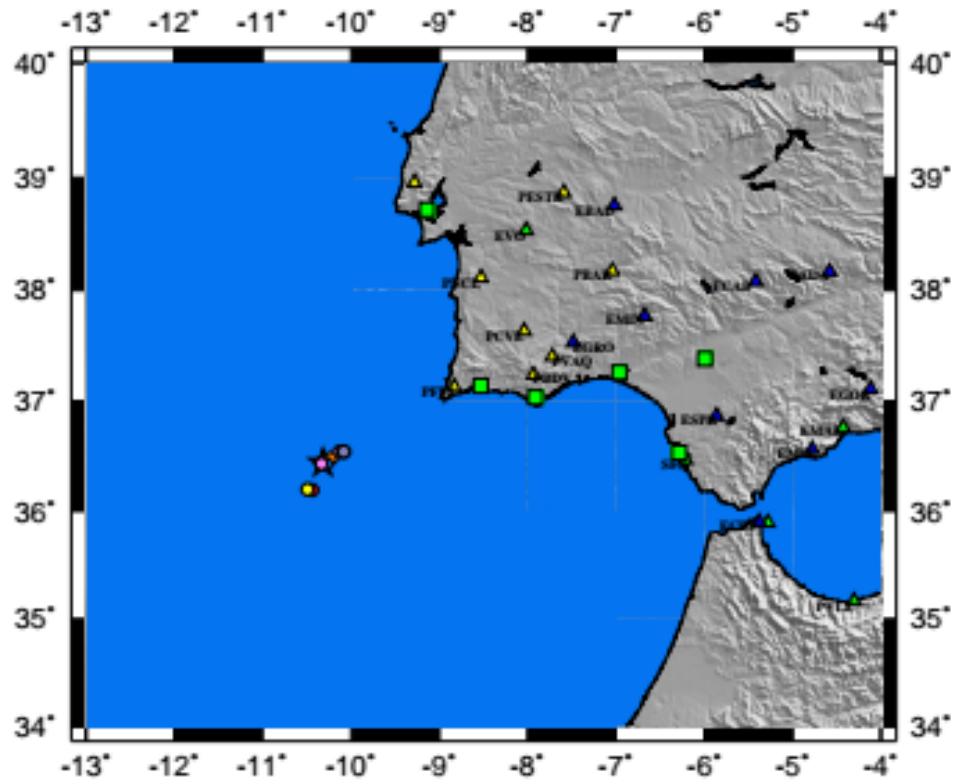
### Pruebas de visualización de datos en los widgets informativos:





**Pruebas de precisión de la imagen y de la conversión de coordenadas geográficas a píxeles de la imagen.**

La imagen del mapa a mostrar debía corresponder a la zona geográfica acotada por las coordenadas que aparecen en el siguiente grid.



**Máxima latitud: 40° N. Mínima latitud: 34° N.**

**Máxima longitud: -3° O. Mínima longitud: -14° O.**

La imagen del mapa acotada a las coordenadas ha sido recortada de una imagen de alta calidad de una proyección que ocupaba una mayor zona geográfica, y se han utilizado las referencias de los accidentes geográficos (con la ayuda de Google Earth y el editor de imágenes GIMP) para que la precisión fuera la mayor posible.

Esta precisión debe ser testada, y para ello hay que usar la función, ya comentada en el apartado de implementación, de conversión de coordenadas a píxeles **coordinatesToPixels**, por lo que esta prueba se ha preparado para una doble vertiente:

1. Testear la precisión de la imagen cortada.
2. Testear la precisión de los cálculos para convertir las coordenadas en grados a píxeles de la imagen.

La prueba se ha diseñado de manera que se impriman en el log de la consola los resultados de los cálculos de conversión coordenadas-píxeles y que en el mapa se visualicen unas marcas que indiquen qué píxeles son los referenciados para así comprobar que la zona marcada en el mapa es la correspondencia real. La marca será una línea a que a modo de flecha

señaladora vaya desde una de las esquinas del mapa hasta el punto marcado (ya que poner solo un punto en los pixeles elegidos no se muestra con suficiente claridad en la imagen).

La prueba ya implementada es la siguiente:

```
/*
*Función privada para probar la precisión de la conversión de
*coordenadas en pixeles de la imagen
*/
void MapWidget::testPixelPrecision(){

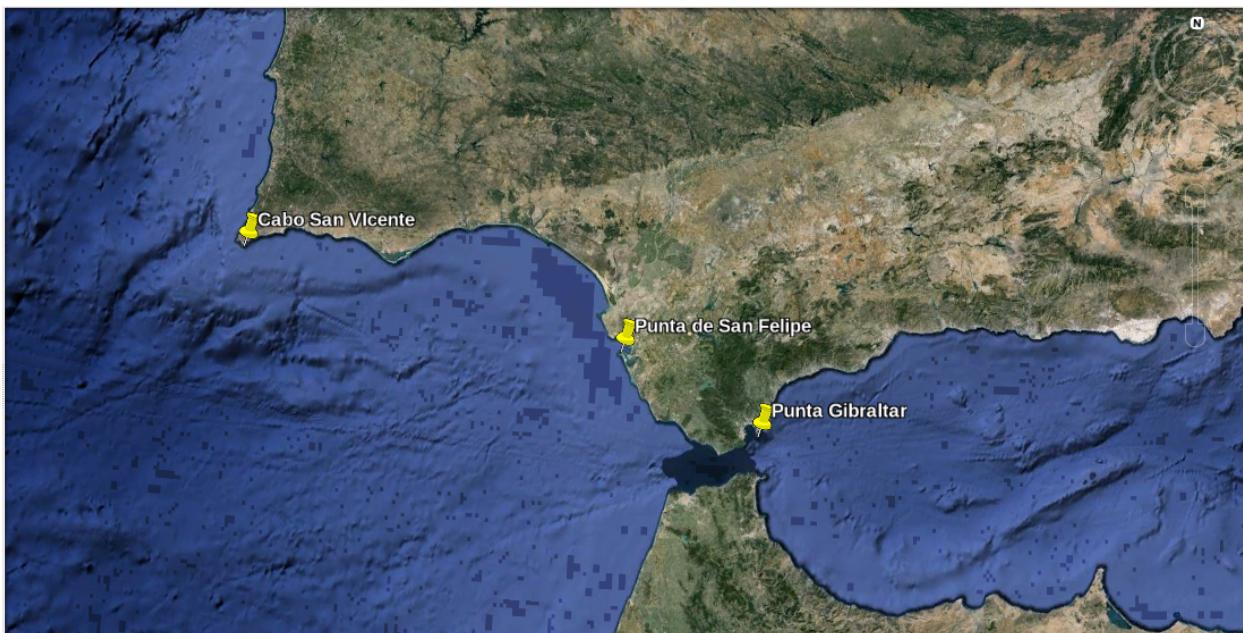
    //Coordenadas Cabo San Vicente
    //36°59'39.61" (36.994336) y -8°-56'-9.60" (-8.936)
    //pintado de líneas
    //desde Cabo San Vicente hasta esquina izquierda arriba
    long double x,y;
    coordinatesToPixels(x,y,36.994336,-8.936);
    std::cout << x << ' ' << y << std::endl;
    mapScene.addLine (x,y,0,0);

    //punta Gibraltar
    //36° 6'34.01" (36.109447) -5°-20'-43.59" (-5.345442)
    //pintado de líneas
    //desde gibraltar hasta esquina derecha abajo
    long double x2,y2;
    coordinatesToPixels(x2,y2,36.109447,-5.345442);
    std::cout << x2 << ' ' << y2 << std::endl;
    mapScene.addLine (x2,y2,
                      mapScene.width (),mapScene.height ());

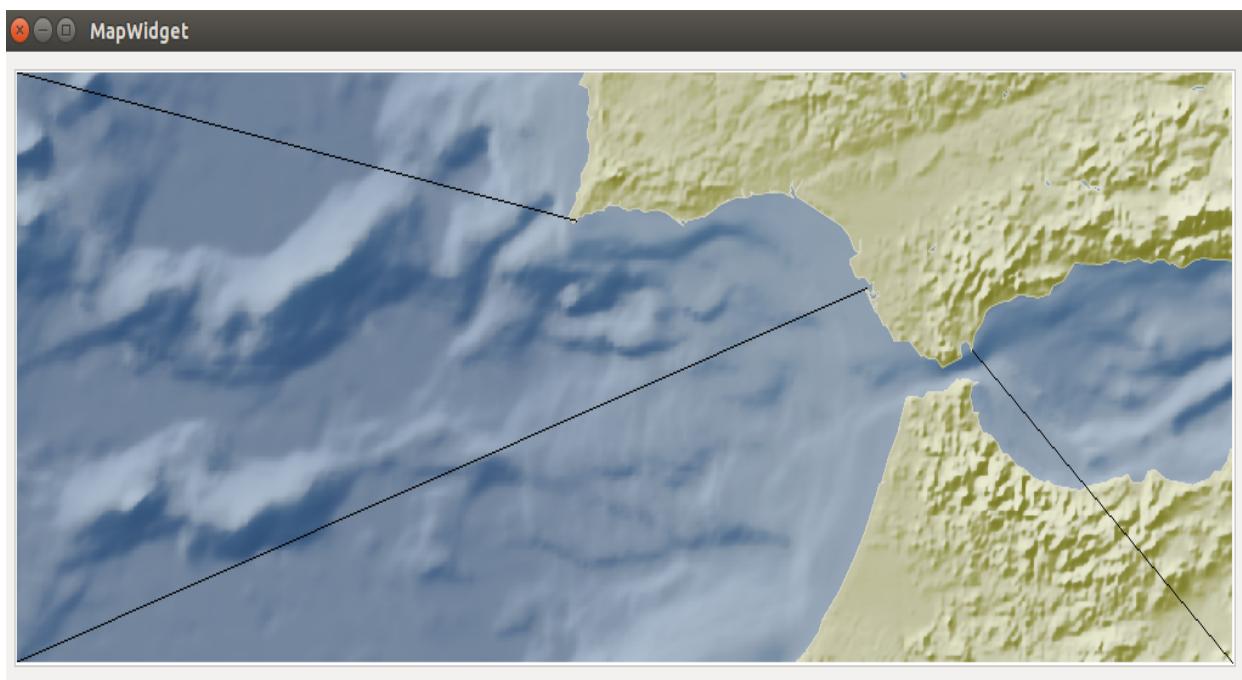
    //Punta San Felipe Cádiz 36°32'16.12" -6°-18'-1.20"
    //pintado de líneas
    //desde Cádiz hasta esquina izquierda abajo
    coordinatesToPixels(x3,y3,36,32,16.12,-6,-18,-1.20);
    std::cout << x3 << ' ' << y3 << std::endl;
    mapScene.addLine (x3,y3,0,mapScene.height ());
}
```

Con la ayuda de Google Earth se puede comprobar si los puntos referenciados coinciden con los que se marcan en nuestro mapa:

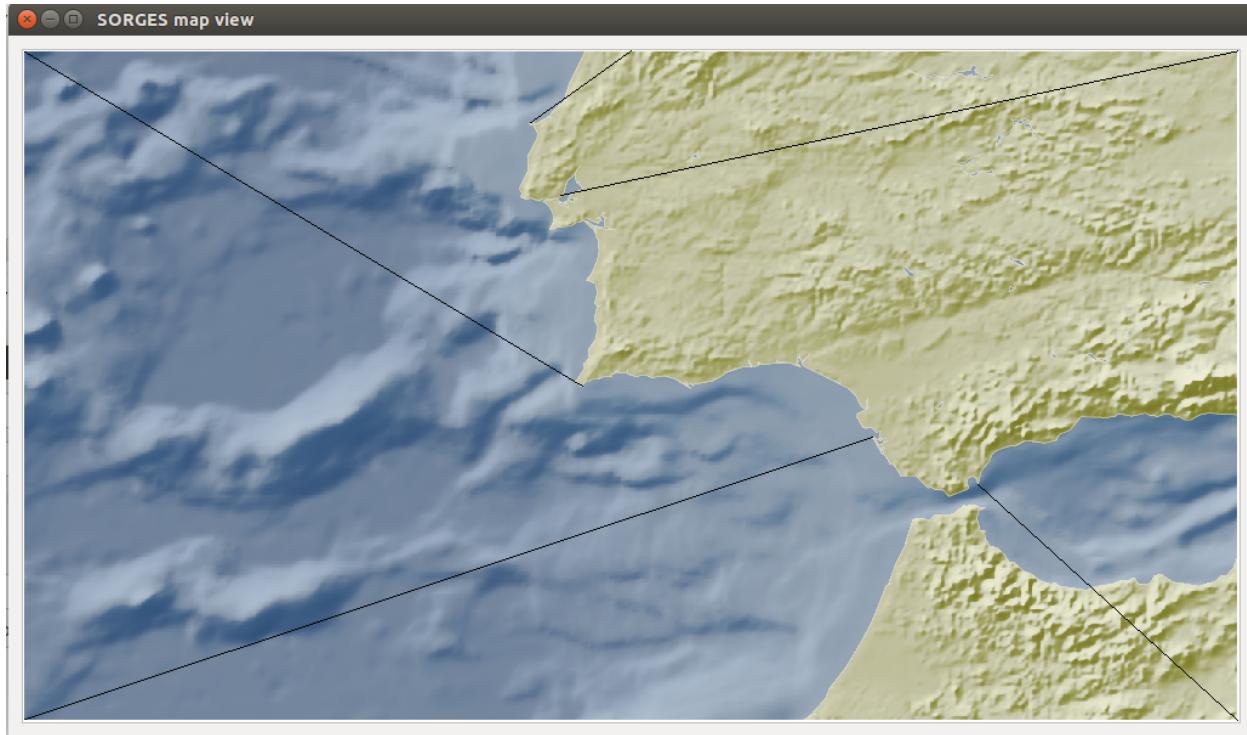
EARTH:



SALIDA DE LA PRUEBA:



Como se puede apreciar, la precisión conseguida prácticamente perfecta.  
En la siguiente imagen se muestra un rango más amplio con alguna prueba más



## 6.2. Integración

### Prueba de origen:

Para probar la integración entre el módulo de datos y el widget gráfico del Mapa, se ha implementado una función en MapWidget la cuál recibirá el objeto *Origin* y sacando sus atributos dibujará en el mapa el epicentro y el primer círculo de expansión.

```
/*
*Función privada para probar la colocación de un Origen en la imagen
*/
void MapWidget::testOrigen(const Origin& myOrigen){
    //Punta san felipe Cádiz 36°32'16.12" -6°-18'-1.20"
    paintOrigin(myOrigen);
}

void MapWidget::paintOrigin(const Origin &origin){
    long double coordX, coordY;
    this->currentOrigin = origin;
    long double radius = calculateRadius();
```

```

coordinatesToPixels(coordX,coordY,currentOrigin.getLatitude(),
                    currentOrigin.getLongitude());

//pintamos las estaciones con su nuevo color
//FUNCION EN PROCESO DE IMPLEMENTACION
//changeStationsColors(currentOrigin.getStations());

QPoint center(coordX, coordY);

// pintamos el primer circulo de expansion.
QRect rect(0,0,2*radius,2*radius);
rect.moveCenter(center);
mapScene.addEllipse (rect,QPen(),
                     QBrush(
                         QColor(R_EPICENTER_FIRST_CIRCLE,
                                G_EPICENTER_FIRST_CIRCLE,
                                B_EPICENTER_FIRST_CIRCLE,
                                T_EPICENTER_FIRST_CIRCLE)));

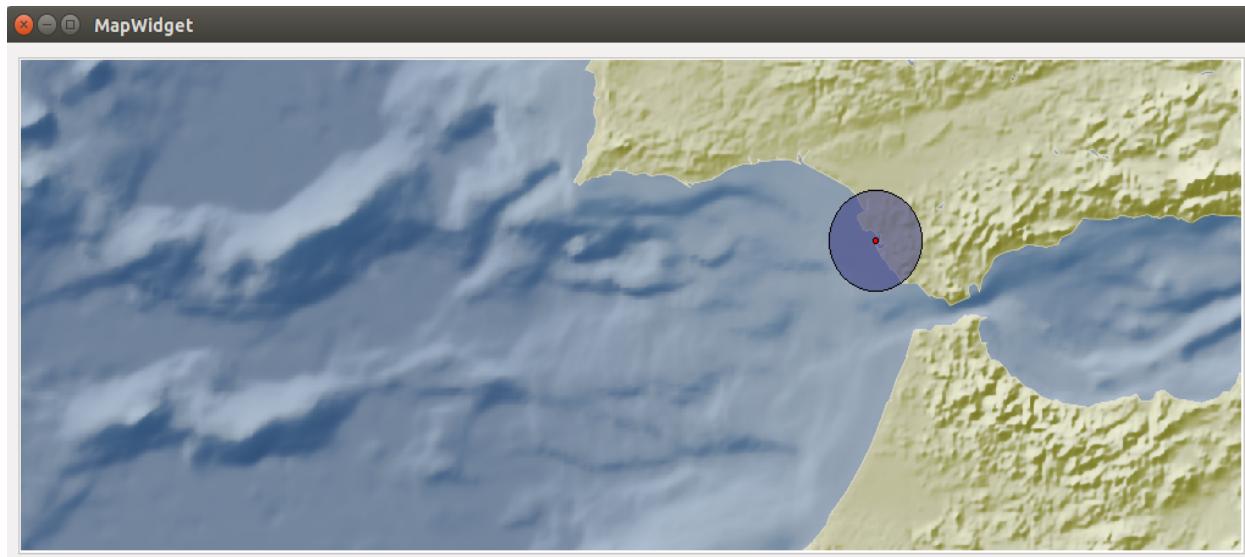
// pintamos el epicentro.
QRect rect2(0,0,2*RADIUS_EPICENTER,2*RADIUS_EPICENTER);
rect2.moveCenter(center);
mapScene.addEllipse (rect2,QPen(),
                     QBrush(
                         QColor( R_EPICENTER,
                                G_EPICENTER,
                                B_EPICENTER,
                                T_EPICENTER)));
}

}

```

Las constantes en mayúsculas definen los colores y el círculo central que va a representar el epicentro y están disponibles en el archivo de configuración *mapdefinition*.

La salida por pantalla es satisfactoria:



### **Prueba de estaciones:**

El módulo de estaciones también está dotado con una serie de pruebas que comprobar el buen funcionamiento del programa, de tal manera que comprobemos que la colocación de la estación es correcta, y el color adoptado. En la prueba, usamos varias coordenadas representativas para testear la precisión de la colocación de estaciones. El resultado como se puede ver en la imagen, es satisfactorio.

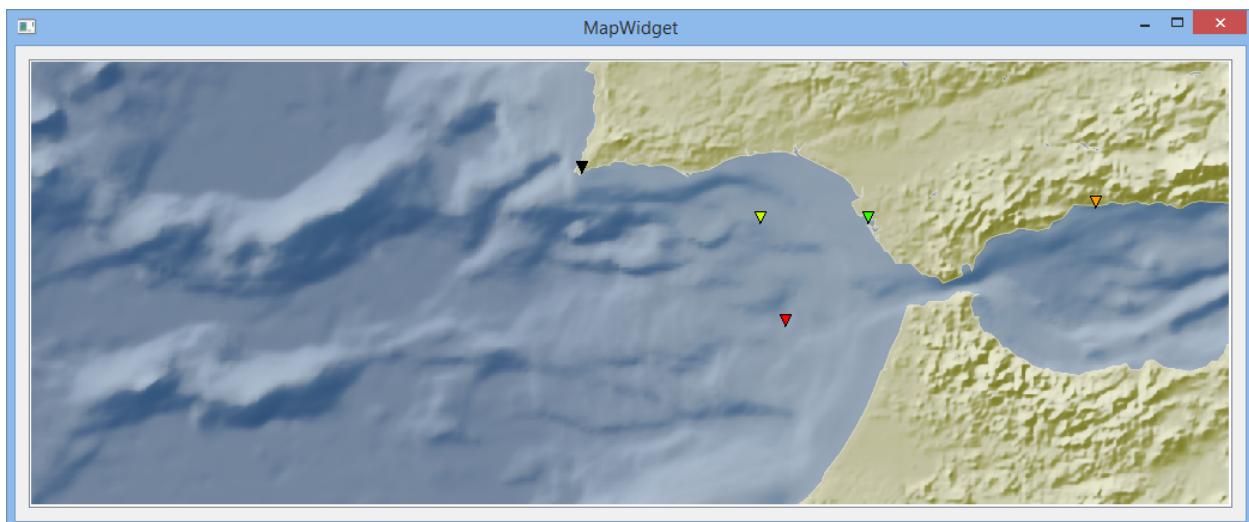
```

/*
 * Funcion privada para probar la colocación de estaciones en la imagen
 */
void MapWidget::testStation(const std::set<Station>& mystations){
    paintStations(mystations);
}

/*
 * Funcion para pintar las estaciones en primera instancia.
 */
void MapWidget::paintStations(const std::set<Station>& stations){
    this->stations = stations;
    for(std::set<Station>::iterator it=stations.begin(); it!=stations.end(); ++it)
        drawStation(*it);
}

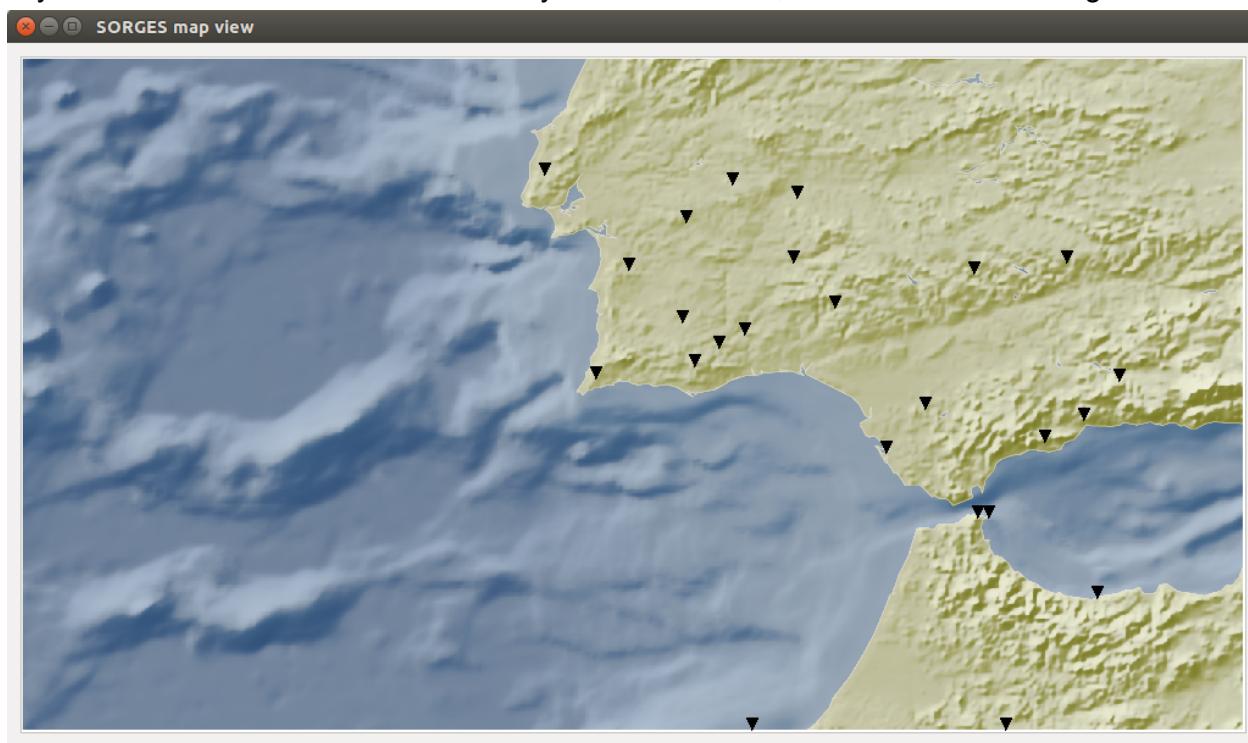
/*
 * Funcion para pintar solo una estación.
 */
void MapWidget::drawStation(const Station& station){
    long double coordX, coordY;
    coordinatesToPixels(coordX,coordY,station.getLatitude(),station.getLongitude());
    QPolygonF Triangle;
    Triangle.append(QPoint(coordX,coordY));
    Triangle.append(QPoint(coordX+STATION_SIZE_X,coordY-STATION_SIZE_Y));
    Triangle.append(QPoint(coordX-STATION_SIZE_X,coordY-STATION_SIZE_Y));
    QGraphicsPolygonItem* pTriangleItem =
mapScene.addPolygon(Triangle,QPen(),QBrush(Station::onSiteAlert[station.getColor()]));
}

```



## Prueba de estaciones desde el fichero proporcionado por ALERT-ES:

Leyendo las estaciones reales de la red y su coordenadas, el resultado sería el siguiente.



### 6.3. Prueba del sistema

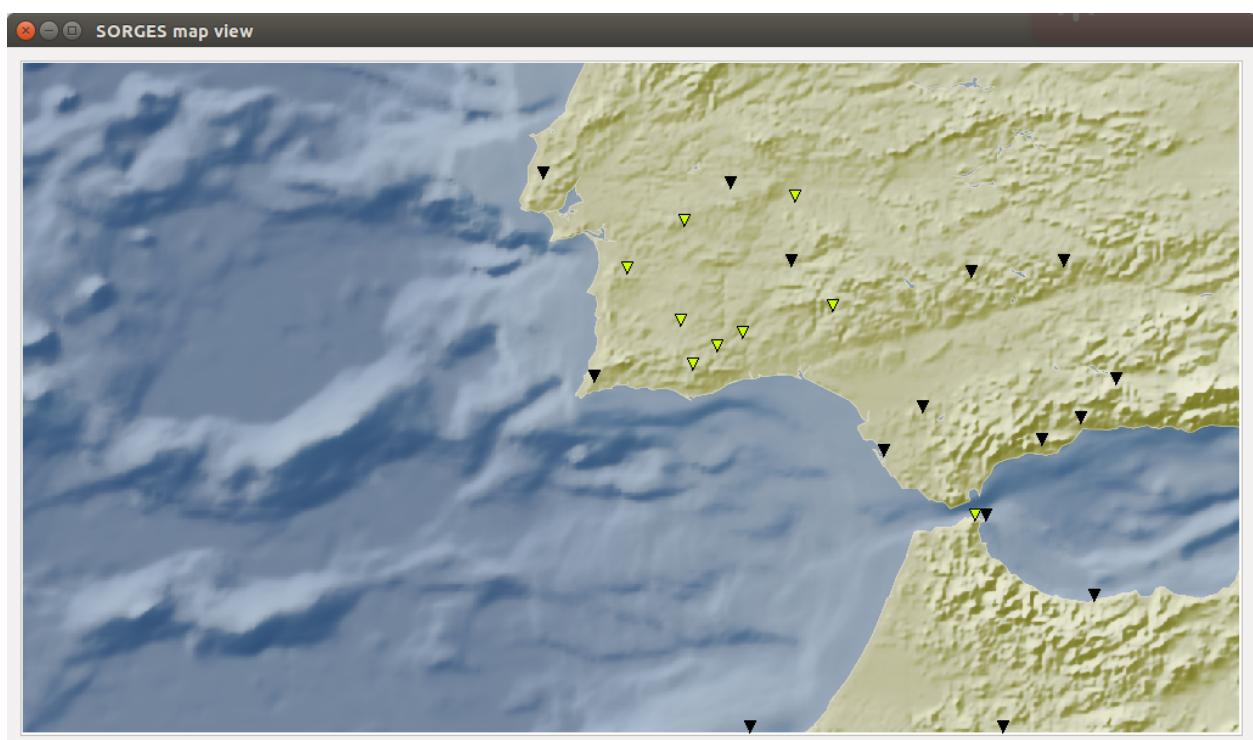
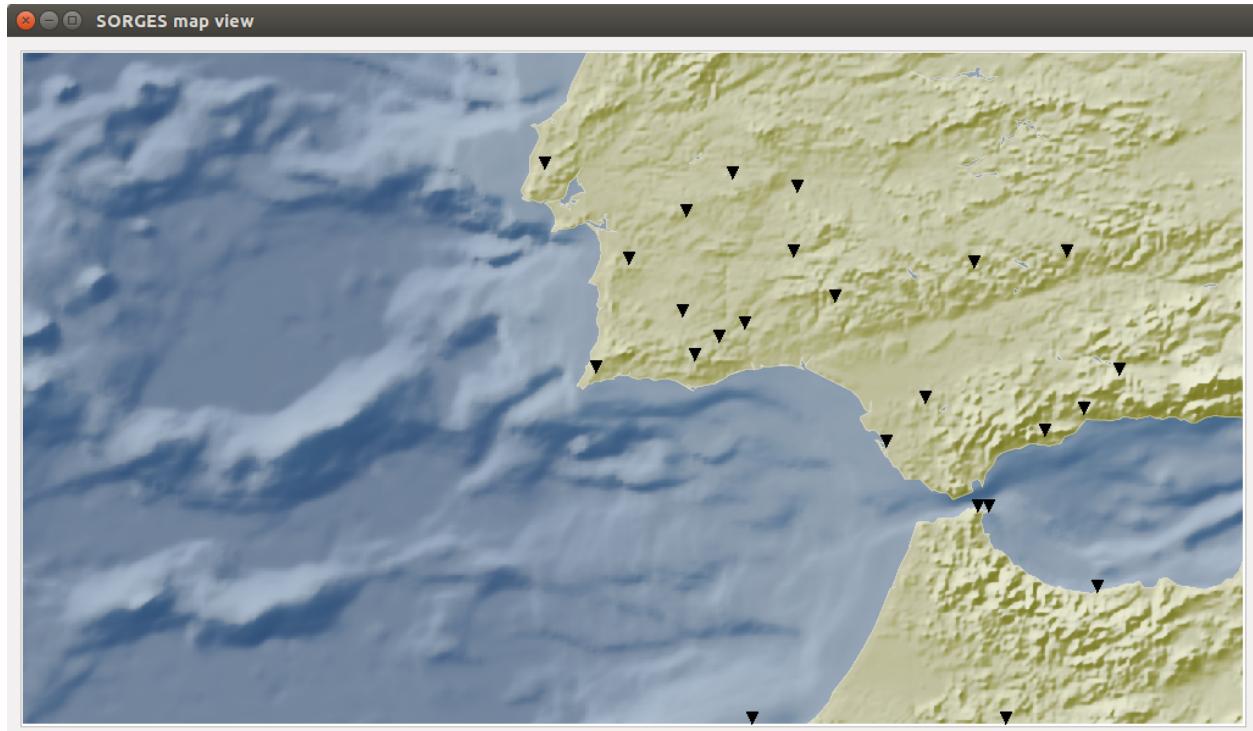
La prueba general del sistema se ha producido mediante la simulación de **todo el proceso que engloba un evento concreto** registrado por la red y que ha sido proporcionado por los técnicos del Real Observatorio de la Armada.

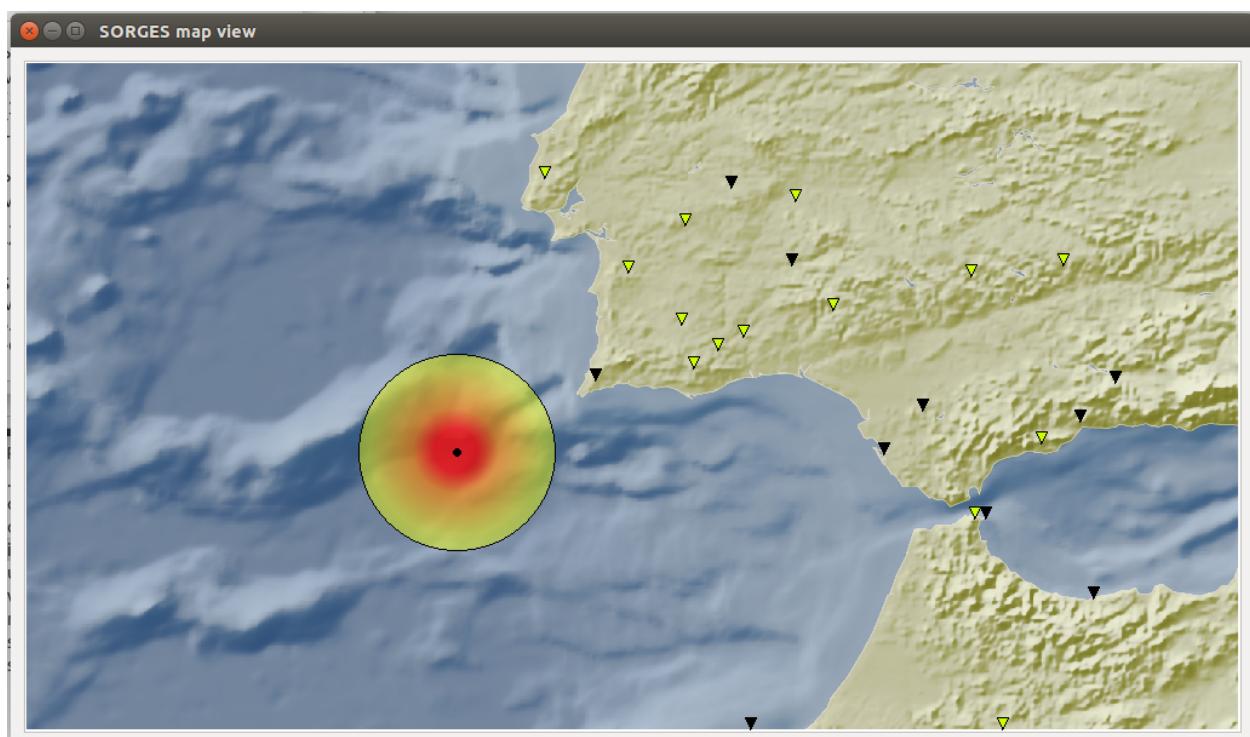
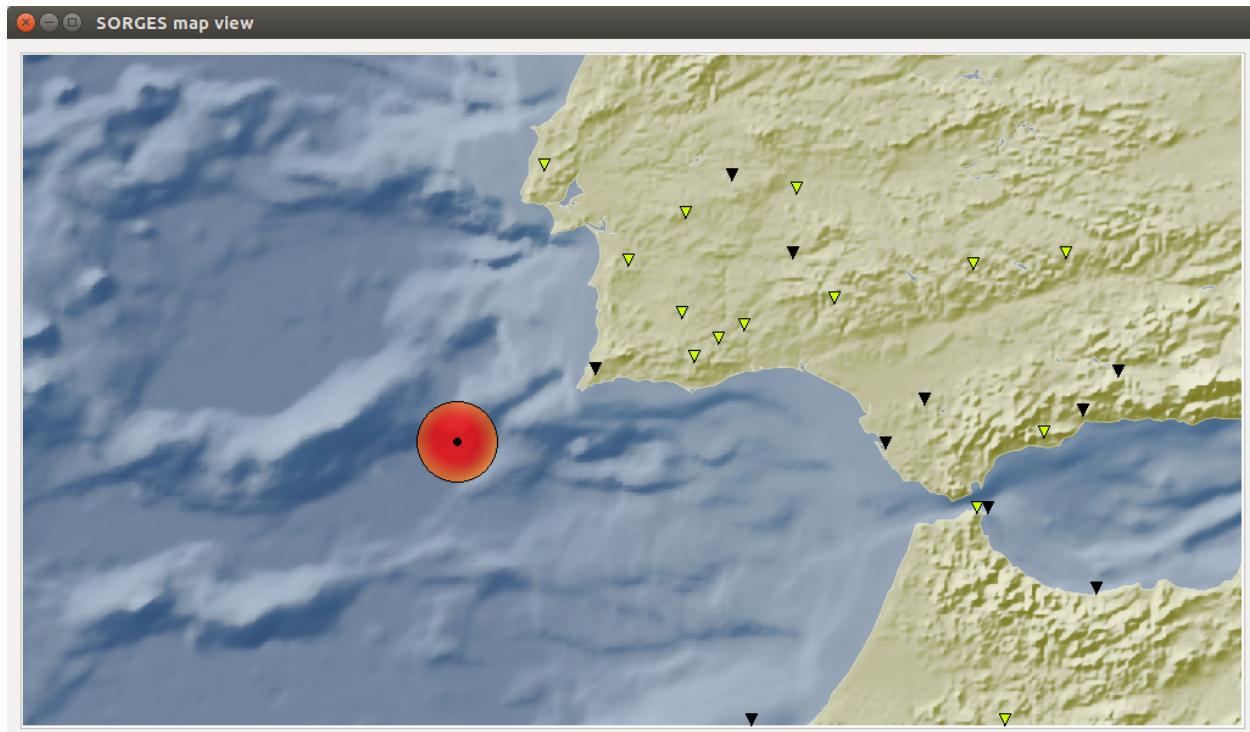
Esta prueba implica:

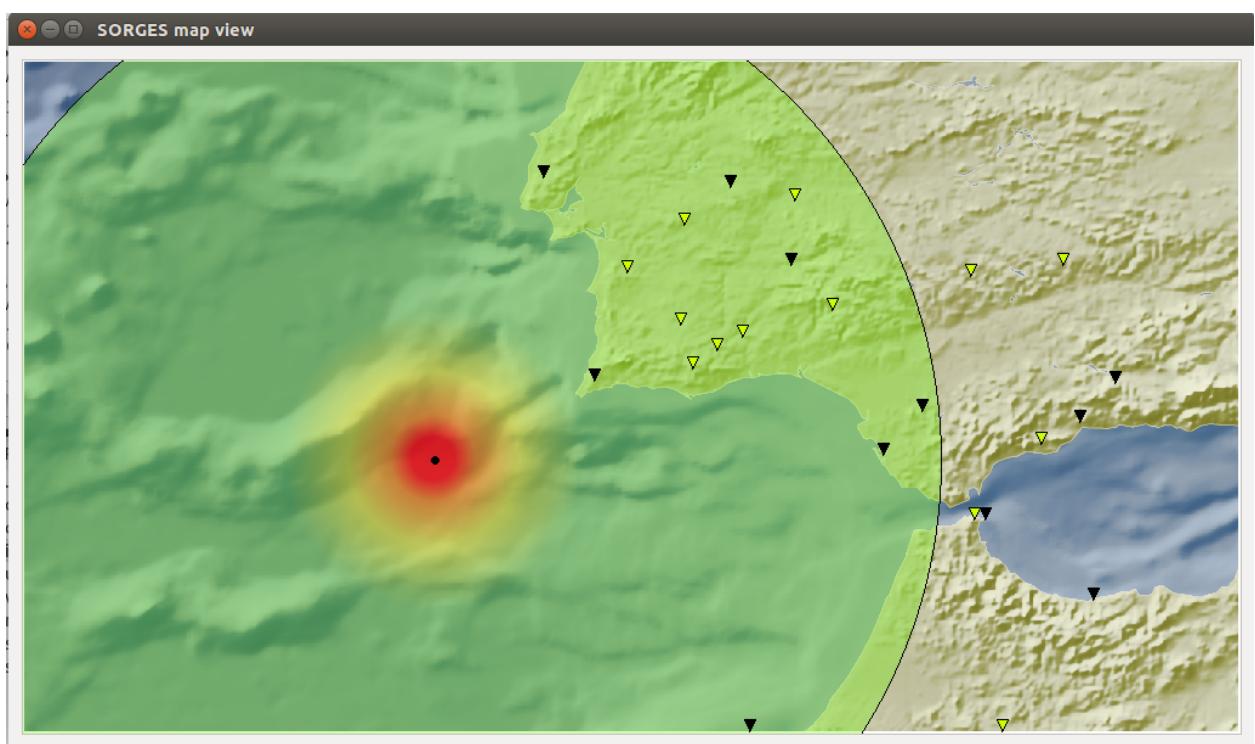
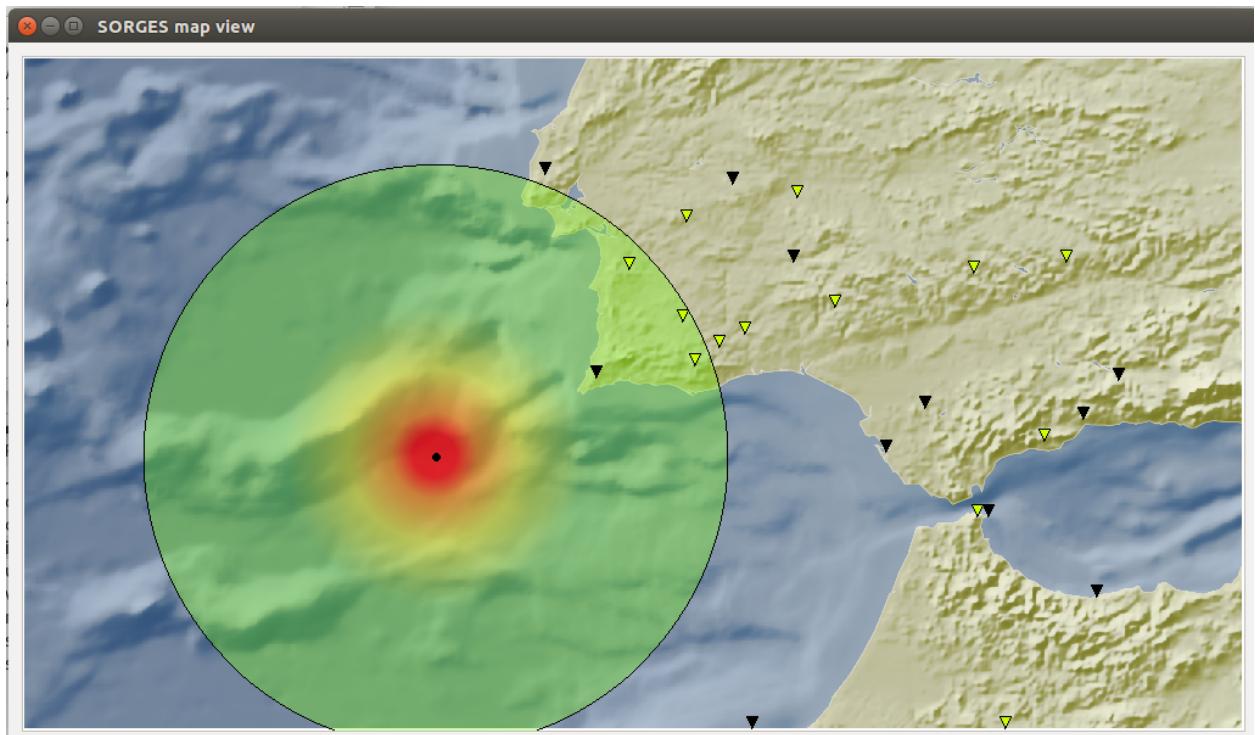
1. Definir la fecha y la hora de la simulación, es decir, la hora del evento requerido para poder buscar sus datos en los logs y xml correspondientes.  
En este caso, se representará un **evento con fecha 20-01-2015 y hora próxima a las 11 de la mañana**.
2. Buscar en la estructura de directorios definida estos ficheros correspondientes. Para evitar búsquedas inconclusas por imprecisiones en la fecha requerida, se ha de controlar que se pueda encontrar el evento en un rango inferior o posterior de fecha y hora, incluyendo límites como pueden ser el primer y último día de mes o la cercanía a la hora 00:00 para el cambio entre días.

La estructura de directorios para buscar los datos de los eventos sigue el siguiente path genérico: **/home/USUARIO/.seiscomp3/log/events/AÑO/MES/DÍA/EVENTO**. La búsqueda es en base una consulta de fechas sobre ficheros xml: se busca en la ruta correspondiente **al día indicado, y al día siguiente y posterior** (incluyendo incluso la posibilidad de que haya que cambiar de directorio de mes y año), y se consulta los datos de todos y cada uno de los eventos en estos directorios. **El evento que sea más próximo en fecha y hora a las indicadas para la simulación será el elegido.**

3. Una vez localizado el evento, hay que sacar de su fichero xml dos fechas: una que indique el **inicio del proceso**, y que coincidirá con la **fecha del primer pick** reflejado, y otra que marque **el fin del proceso**, y que será una de las fechas reflejadas en la información del evento como **modification time**. Con estas dos fechas, ya tenemos el rango en el cual tenemos que buscar los datos en los ficheros de log.
4. Procesar los ficheros de logs: el fichero de picks y el fichero de orígenes de alertes. Estos se encuentran en las rutas genéricas  
**/home/USUARIO/.alertes/log/picks/scalertes\_picks.log y**  
**/home/USUARIO/.alertes/log/origins/scalertes\_origenes.log**. Dentro de estos logs, hay que extraer toda la información contenida entre el rango de fechas de inicio y fin antes obtenidas.
5. Una vez esta información es extraída, se divide en bloques que en el log aparecen con la misma fecha de llegada, y se almacena una lista de bloques en la que cada uno de ellos tendrá también la información de la diferencia de tiempo existente con su predecesor.
6. Cuando se tiene toda esta información estructurada, se planifica la simulación para volcar todos estos datos en los ficheros de simulación. Estos ficheros de simulación funcionan como si de un tiempo real se tratases: cuando se actualiza con nueva información, se envía una señal que es recogida por el procesador de datos, que crea las estructuras y las manda al módulo gráfico para su representación.  
El planificador de la simulación controla un temporizador que va volcando datos a los ficheros mencionados siguiendo los intervalos de tiempo indicados en los bloques, así va enviando información de picks, orígenes y el evento al finalizar el proceso.
7. El módulo gráfico recibirá las estructuras conforme se actualicen los datos volcados a los ficheros de simulación, y representará el proceso, cambiando de color las estaciones según los picks llegados y su alerta y pintando los epicentros y expansiones de los distintos orígenes y el evento. Una muestra de algunos pasos de este proceso se puede ver a continuación.







## **7. Conclusiones**

### **7.1. Aspectos generales**

Durante el desarrollo del proyecto se han tenido dificultades en el aprendizaje de la terminología de los eventos sísmicos, pues es un tema inédito para nosotros, pues en el argot científico existen multitud de tecnicismos que no conocíamos.

Haciendo referencia a la implementación del sistema, hemos encontrado algunos aspectos que nos han acarreado problemas, como son:

- obtención del flujo de datos y su asimilación.
- procesamiento de datos de ficheros sin estructura definida.
- estudio del IDE Qt de desarrollo de aplicaciones con interfaz gráfica de usuario.
- trabajar con el sistema de señales y slots, y con los temporizadores para los distintos eventos temporales.

Mediante este proyecto, nos hemos familiarizado en la relación con el cliente y nos ha acercado a nuestra vida laboral futura. El hecho de que los requisitos estén en constante movimiento en la vida real es una prueba más, y se aprenden experiencias en las cuales hay que lidiar con cambios rápidos o correcciones de funcionalidades debidas a falta de entendimientos puntuales que pueden surgir.

Finalmente, con este proyecto estamos poniendo en práctica todo lo aprendido durante la carrera, incluso aprendido nuevos aspectos de programación y uso de nuevas herramientas nunca antes vistas por nosotros. También hemos aprendido a emprender un proyecto desde cero, realizar un trabajo de investigación y análisis de soluciones, fomentar nuestras habilidades autodidactas y desarrollar un producto completo.

El trabajo en equipo y cómo se ha de llevar debe ser una de las conclusiones principales sobre las que debemos reflexionar, ya que al ser un proyecto de cierta entidad ha requerido una coordinación precisa y muchos momentos de sacrificio.

### **7.2. Conocimientos adquiridos**

Este proyecto nos ha ayudado en muchos aspectos, tanto de práctico como empresarial.

En el apartado práctico, el conocimiento de nuevas herramientas como Qt o Git son de gran ayuda ya que su uso en el futuro es muy conveniente. Los recursos gráficos usados

complementan nuestra enseñanza ya que nunca antes habíamos trabajados con interfaces gráficas.

Además, la envergadura del proyecto ha hecho que el esfuerzo organizativo y de planificación haya sido considerable, cosa que en un futuro nos va a servir como experiencia.

En cuanto al apartado empresarial, el trato con clientes reales, las peticiones, los cambios y las propuestas.

### 7.3. Futuro del proyecto

El futuro inmediato de este proyecto es la prueba de **implantación en los sistemas del Real Observatorio de la Armada**, para lo cuál habrá que depurar su funcionamiento una vez esté interactuando con el entorno en el que va a ejecutarse.

Además, algo que se ha planteado para una ampliación de SORGES es la **conexión con ALERT-ES vía socket para la obtención de datos**: el sistema podría ser capaz de conectarse con el sistema ALERT-ES a través de un socket para recibir la información de estaciones y orígenes.

## 8. Referencias y Bibliografía

[1] Documentación de c++. <http://www.cplusplus.com/>

[2] Documentación de Qt. <http://doc.qt.io/>

[3] Wiki de Qt. <http://qt-project.org/wiki>

[4] Alert-es. <http://www.alertes.roa.es/>

[5] Seiscomp 3. <http://www.seiscomp3.org/>

[6] GNU GPL: <http://www.gnu.org/licenses/licenses.es.html>

# ANEXOS

## A. Herramientas utilizadas



**QtCreator:** Entorno de programación para Qt.



**Repositorios Git:** Control de versiones del proyecto.



**GitHub:** Servicio Host para el control de versiones de Git.



**Diagramas Cacoo:** Herramienta de creación de diagramas Online.



**Hangouts de Google:** Servicio web para videollamadas y chat.



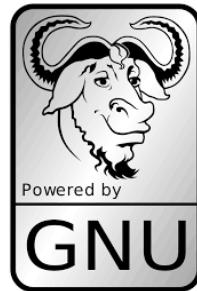
**Google Drive:** Herramientas ofimáticas online



**Dropbox:** Host para compartir archivos.

## B. Publicación de la aplicación

El equipo desarrolladores de SORGES, nos decantamos publicar el proyecto bajo la licencia GPL de GNU. La Licencia Pública General de GNU, se usa para la mayoría de los programas de GNU y para más de la mitad de los paquetes de software libre. La última es la versión 3.



GNU GPL es la licencia más ampliamente usada en el mundo del software y garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

En uso puramente privativo (o interno), sin ventas ni distribuciones implicadas, el software puede ser modificado sin liberar el código fuente pero, de lo contrario, el código fuente y cualquier cambio realizado en él debe estar disponible para los usuarios, ya que en este caso los derechos del usuario están protegidos por copyleft. De esta forma, las aplicaciones instaladas en sistemas operativos bajo licencia GPL como Linux.

## C. Manual de instalación

### a. Entornos Linux

#### i. Producto compilado.

El programa compilado no necesita instalación específica. Al obtener el archivo comprimido y descomprimirlo encontrará la siguiente estructura.



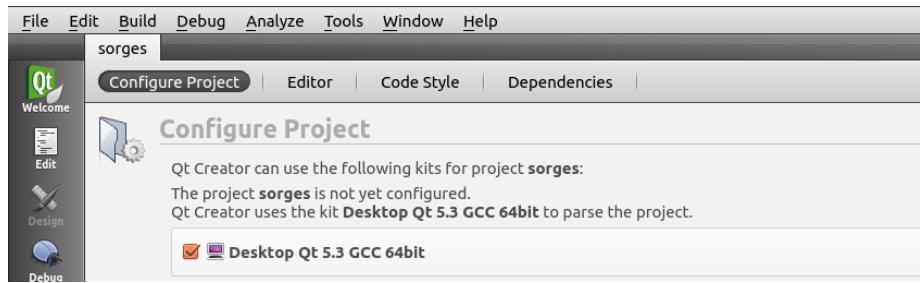
1. *sorges*: es el ejecutable de la aplicación. Con doble click iniciará el modo a tiempo real, para la simulación necesita ejecutarse con los parámetros adecuados desde la línea de comandos.
2. *config*: en este directorio se encuentra el archivo de configuración de la aplicación, donde se debe especificar los paths a los que hay que acceder para extraer los datos (*filepaths*). También se especifican los paths para ficheros de simulación (*simulationpaths*) aunque estos últimos valores no hay que modificarlos puesto que se modifican en tiempo de ejecución para apuntar al directorio *simulationFiles*.  
NOTA: por defecto, el path de acceso a los ficheros de alertes y seiscomp viene reflejado ya en el archivo. Este path se toma desde */home/USUARIO/.seiscomp3* y */home/USUARIO/.alertes*.
3. *simulationFiles*: directorio donde se almacenan los ficheros que se utilizan para volcar los datos sísmicos secuencialmente en el modo simulación.
4. *backup*: directorio donde se almacenarán en formato xml volcados de todos los orígenes/eventos y estaciones que se representen en la ejecución.

#### ii. Código fuente.

Para compilar los ficheros fuente, recomendamos el uso del entorno *QtCreator* ya comentado en esta documentación, ya que se han usado sus generadores de *qMake* y *Makefiles* para la compilación.

Para abrir el proyecto en *QtCreator* tan solo hace falta abrir con esta herramienta el archivo *sorges.pro*, que cargará el resto de ficheros.

La compilación se debe hacer con la configuración de compilación de C++, ya sea en 64 ó 32 bits dependiendo de la máquina.



### b. Entornos Windows y MAC-OS.

No se garantiza el correcto funcionamiento de la aplicación en estos sistemas, ya que puede haber incompatibilidades de configuración.

## D. Manual de usuario

El proyecto esta orientado a un público con unos conocimientos mínimos de informática, así como usuarios de Linux, ya que el software no es compatible con el sistema operativo de Microsoft ni Apple.

El modo **Tiempo real** se puede ejecutar desde la carpeta del programa de dos formas, clicando en el ejecutable o desde consola ejecutando el comando **./sorges** o bien **./sorges realtime**.

A continuación el programa se ejecutará de forma indefinida en la que procesará la información que vaya llegando a los ficheros especificados en el archivo de configuración, hasta que el usuario decida cerrar la aplicación.

Para el modo **Simulación** hay que ejecutar el comando indicando la fecha del evento que se quiere simular: **./sorges simulation yyyy-MM-dd hh:mm:ss**.

La fecha especificada debe seguir ese formato. El programa iniciará la carga de datos del evento especificado y comenzará la simulación de todo el proceso. La pantalla se limpiará a los 5 minutos de la finalización del evento.

## E. Licencia

# GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this

License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to

thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## **2. VERBATIM COPYING**

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## **3. COPYING IN QUANTITY**

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin

distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## **5. COMBINING DOCUMENTS**

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any

sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## **6. COLLECTIONS OF DOCUMENTS**

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## **7. AGGREGATION WITH INDEPENDENT WORKS**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## **8. TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these

Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## **9. TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

## **10. FUTURE REVISIONS OF THIS LICENSE**

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to

the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

## 11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.