Experiencia de colaboración en el desarrollo de GIS opensource (QGIS, GDAL-OGR y GeoServer-GWC).

Alvaro Huarte Sanz[(1)]

[(1)] Departamento de Sistemas Información Territorial de Tracasa (www.tracasa.es), ahuarte@tracasa.es.

RESUMEN

Este artículo presenta la experiencia personal obtenida en la colaboración al desarrollo de plataformas GIS opensource tan maduras como QGIS, GDAL-OGR o GeoServer-GWC.

Dicha necesidad en principio nos vino marcada en primera instancia para cubrir desde Tracasa (www.tracasa.es) ciertas funcionalidades dentro del marco del proyecto SITNA (Sistema de Información Territorial de Navarra) y a través de su IDE, IDENA (http://idena.navarra.es). Posteriormente y ya con fines puramente voluntarios, la implementación de determinadas funcionalidades en alguno de estos productos.

Se describirán los requisitos técnicos que se han encontrado para el desarrollo colaborativo (IDE's, gestión de repositorios, gestión de branches), así como el feedback obtenido en la implementación de issues/features de estos productos y de la interrelacción con la comunidad existente, y muy madura, de gestores de producto, desarrolladores y testers.

Esta ponencia pretende demostrar con nuestra experiencia, aun siendo neófitos en dichos proyectos, la excelente valoración positiva de dicha colaboración "opensource" y animar al resto de la comunidad a ser proactivo en su desarrollo.

Palabras clave: *QGIS, GDAL-OGR, GeoWebCache (GWC), github, Open source, desarrollo colaborativo.*

INTRODUCCIÓN

De la Wikipedia, el término "Open Source" [1] (código abierto en castellano) es la expresión con la que se conoce al software distribuido y desarrollado libremente. Más resumido y a la vez esclarecedor no se puede describir el concepto. En general, y salvando algunos tipos de licenciamientos más restrictivos, el software "opensource" se desarrolla y distribuye sin costes de utilización para los usuarios ni impedimentos en el empaquetado de nuevos aplicativos basados en él. La comunidad puede libremente leer, distribuir e incluso desarrollar nuevas funcionalidades en estas aplicaciones informáticas.

Hace ya tiempo que este tipo de software está entre nosotros, y aunque en un principio el "opensource" estaba asociado con un halo romántico y a la vez transgresor del software propietario, a día de hoy el software "abierto" es visto como una perfecta alternativa seria y profesional y con funcionalidades tan buenas o mejores que sus equivalentes en el software propietario.

El grado de madurez de algunas de estas aplicaciones es tal que la comunidad ahora puede cubrir las necesidades de sus proyectos eligiendo libremente, y nunca mejor dicho, entre aplicaciones "opensource" o privativas. A estas alturas no tiene sentido polemizar sobre qué es lo mejor, el caso de uso nos lo definirá para nuestras necesidades. Las aplicaciones abiertas a priori son más atractivas para los usuarios, pero siempre deberemos estar atentos a que su elección no sea determinada por la moda "opensource" y caigamos en los mismos prejuicios históricos pero vistos desde justo el otro extremo.

En el sector de las aplicaciones que gestionan la Información Geográfica, podemos congratularnos de la existencia de un gran abanico de desarrollos "opensource" muy maduros y con potentes capacidades, similares en muchos casos a las de sus equivalentes privativos, y en ocasiones incluso superiores. Disponemos de varias aplicaciones desktop GIS (QGIS[2], gvSIG[3], Grass[4], uDIG[5], …) con grandes capacidades de gestión y procesamiento de datos geográficos, servicios WEB con potente funcionalidad geográfica y además conforme a estándares OGC (GeoServer[6], MapServer[7] …), potentes clientes WEB mapping para consulta e incluso edición de la información geográfica en exploradores WEB o en dispositivos móviles (OpenLayers[8], Leaflet[9], …), componentes programables como las GDAL-OGR[10], e incluso paquetes listos para instalar con completas arquitecturas de aplicaciones que implementan la funcionalidad geográfica conforme a estándares (Boundless[11], Deegree[12], …).

Estas últimas referencias nos sirven también para desmitificar que el "opensource" no encaja en las empresas y que no se puede "vivir de ello". Tenemos que ser conscientes de que este software alguien lo desarrolla; en algunos casos la comunidad que lo empuja es mayoritariamente desinteresada, en otros casos existen ya empresas o consorcios de empresas que también colaboran, e incluso casos en que ellas mismas los lideran. El retorno que obtendrán estas empresas serán en créditos publicitarios, marketing, en que son organismos, públicos o privados, comprometidos con la democratización de las tecnologías, en el soporte y/o personalización del software base, o … que cada uno aporte su motivo.

Lo bueno de todo esto es que el usuario final es el beneficiado, el “opensource” nos ofrece una opción muy interesante para cubrir nuestras necesidades profesionales, nos permite mejorarlas, o como mínimo platearnos mejorarlas y puede convivir en armonía con el “otro” software si así viene requerido.

Este artículo pretende detallar nuestra experiencia en la implementación de nuevas funcionalidades en algunas de estas aplicaciones “opensource” con funcionalidad geográfica (QGIS, GDAL-OGR y GeoWebCache-GWC) para cubrir en principio las necesidades de algunos de nuestros proyectos, y posteriormente ya como partícipes “libres” conocedores de que estos desarrollos necesitan del empuje de todos y de que todos en la medida de nuestras posibilidades somos corresponsables y beneficiarios de sus resultados.

ARQUITECTURA DEL DESARROLLO COLABORATIVO

Como se ha comentado en el apartado anterior procederemos a describir en un tono informal los pasos que seguimos para colaborar en algunos entornos “opensource” ya existentes y con muchos años de andadura.

El primer paso, y fundamental, es perder el miedo, debemos salvar nuestros propios prejuicios. Todos podemos en la medida de nuestras posibilidades colaborar de una forma una u otra en uno de estos entornos. Los proyectos “opensource” están sustentados en muchos aspectos y nosotros seguro sabemos de alguno. Hay roles para desarrollar el código de las aplicaciones, pero no menos importante, hay roles para documentar las capacidades del software (aspecto vital que en muchos casos no se valora como se debe), traducciones a diferentes idiomas de las herramientas, testeadores de nuevas funcionalidades y de las llamadas regresiones (nuevos “bugs” en herramientas ya existentes pero que surgen por el retoque del código para la implementación de nuevos componentes)… Seguro, seguro, que sabemos de alguno de esos aspectos ¿no?, las nuevas manos serán siempre bienvenidas así que manos a la obra.

En nuestro caso, nuestra primera experiencia vino de la mano de GeoWebCache (GWC) [13] implementando nuevo código para cubrir las necesidades de un determinado proyecto, para luego implementar o dar soporte para nuevas herramientas genéricas definidas en el “road” general de GDAL-OGR y QGIS.

Con el objeto de focalizar el mensaje voy a describir mi experiencia en la colaboración en el desarrollo de ciertas funcionalidades presentes en QGIS 2.2 “Valmiera” pero esto es perfectamente extrapolable, evidentemente con sus matices específicos, a las otras aplicaciones comentadas, o a muchos “opensource” que pululan por Internet.

QGIS (anteriormente llamado también Quantum GIS) es un Sistema de Información Geográfica (SIG) de código libre para plataformas GNU/Linux, Unix, Mac OS y Microsoft Windows. Era uno de los primeros ocho proyectos de la Fundación OSGeo y en 2008 oficialmente graduó la fase de incubación. Permite manejar formatos raster y vectoriales a través de las bibliotecas GDAL y OGR, así como gestionar bases con almacenamiento geográfico.

Para la colaboración en el proyecto, en primer lugar es primordial aterrizar en el entorno y estudiar la infraestructura publicada del producto para entender los pasos necesarios para poder colaborar. En el proyecto QGIS la infraestructura está muy bien definida, y así debería serlo para una aplicación tan madura y en la que colaboran personas y empresas de todo el mundo, y su punto de inicio no es otro que la página del producto:

http://www.qgis.org/es/site

Navegando por su portal más en detalle nos encontraremos con la página principal de colaboración en la aplicación estructurada para las diferentes áreas del proyecto, sean aspectos de documentación, traducción o desarrollo puro y duro de QGIS o de plugins para QGIS:

http://www.qgis.org/en/site/getinvolved/index.html

Para el desarrollo de código en QGIS debemos formarnos en las herramientas que se nos ofrecen para integrarnos en su infraestructura. Existe un repositorio central de código fuente, documentación de cómo bajar código a nuestras máquinas, de cómo conseguir compilar el código, de cómo subir los cambios que hayamos sido capaces de desarrollar, hay listas de "bugs" o nuevas funcionalidades que los usuarios añaden y listas de correo bien estructuradas en varios roles; para desarrolladores, documentadores, aspectos visuales de la aplicación, notas de usuarios…

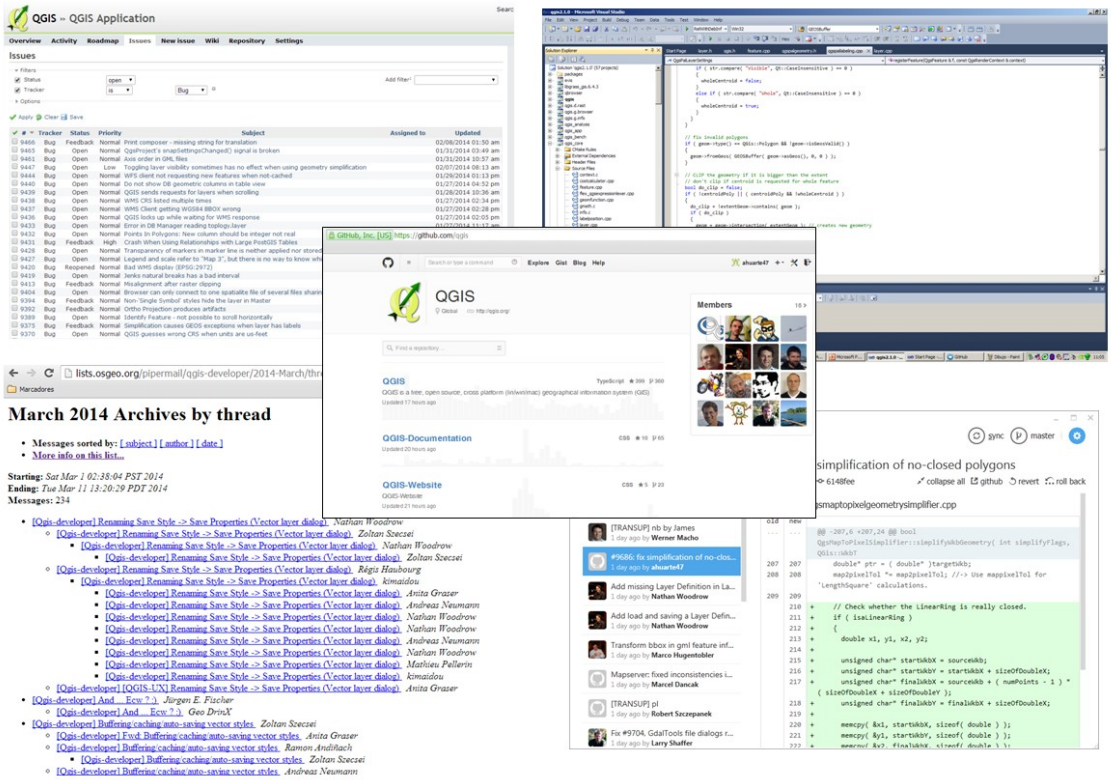

Figura 1: Abanico de recursos y herramientas para el desarrollo de QGIS.

Pasamos pues a describir estos recursos.

Repositorio de código fuente

El código de la aplicación, y otros recursos relacionados con QGIS, se encuentra alojado en “github” [14]:

https://github.com/qgis/QGIS

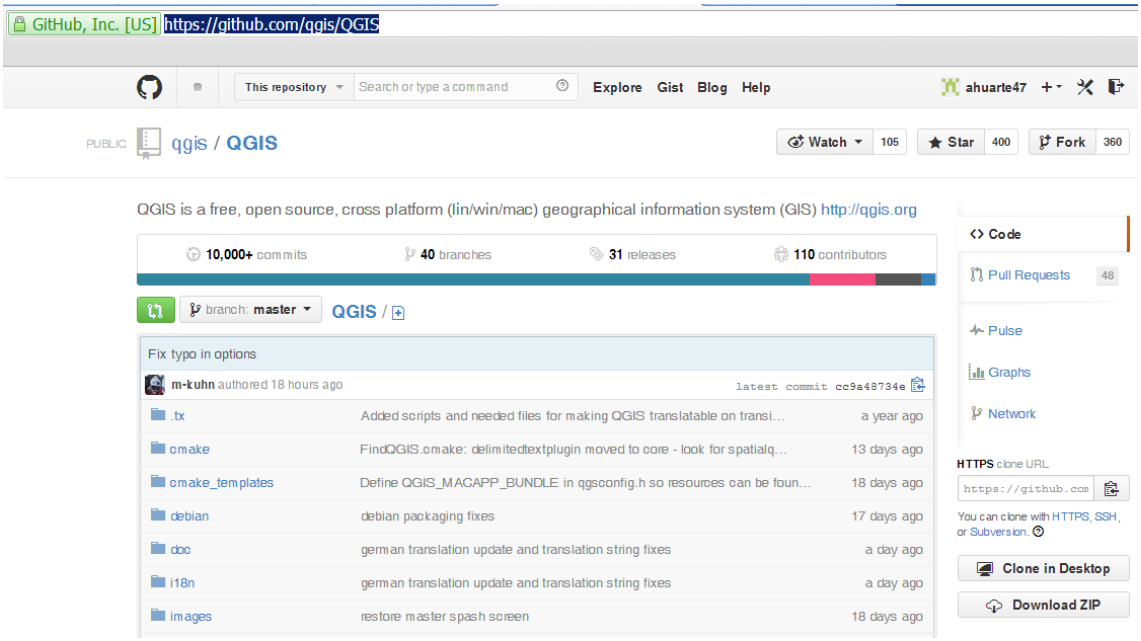

Figura 2: Página principal del repositorio de QGIS desktop en github.

Github es un sitio web gratuito que sirve para alojar proyectos de todo tipo y que utiliza un sistema de control de versiones. Los usuarios pueden crear nuevos repositorios, bajarse ficheros agrupados por versiones y publicar nuevas versiones de ellos, o como mínimo notificarlas para que sean evaluadas y finalmente aceptadas por los administradores del proyecto.

Pero antes de seguir con QGIS e intentar teclear una línea de código, es importante detallar el flujo de trabajo.

El primer paso es registrarse, si no lo estamos ya, en “github”. Este trámite nos permitirá gestionar los repositorios alojados en este sitio WEB, y en nuestro caso el de QGIS.

https://github.com/join

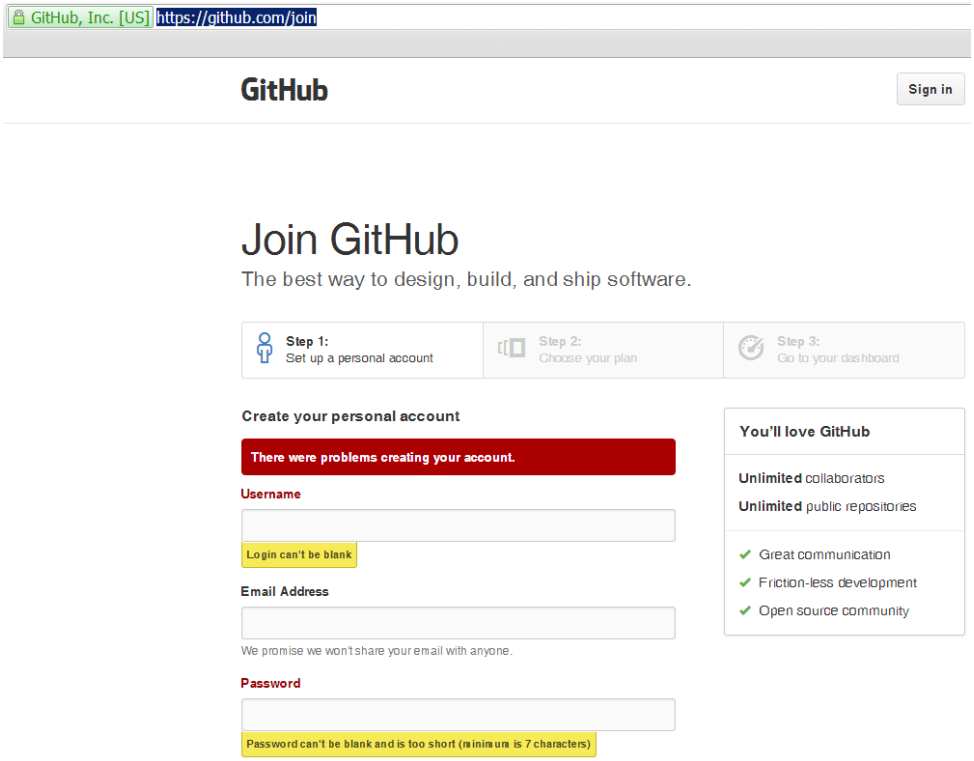

Figura 3: Página de registro en github.

Tras los pertinentes datos a introducir. Ya tenemos nuestro usuario:

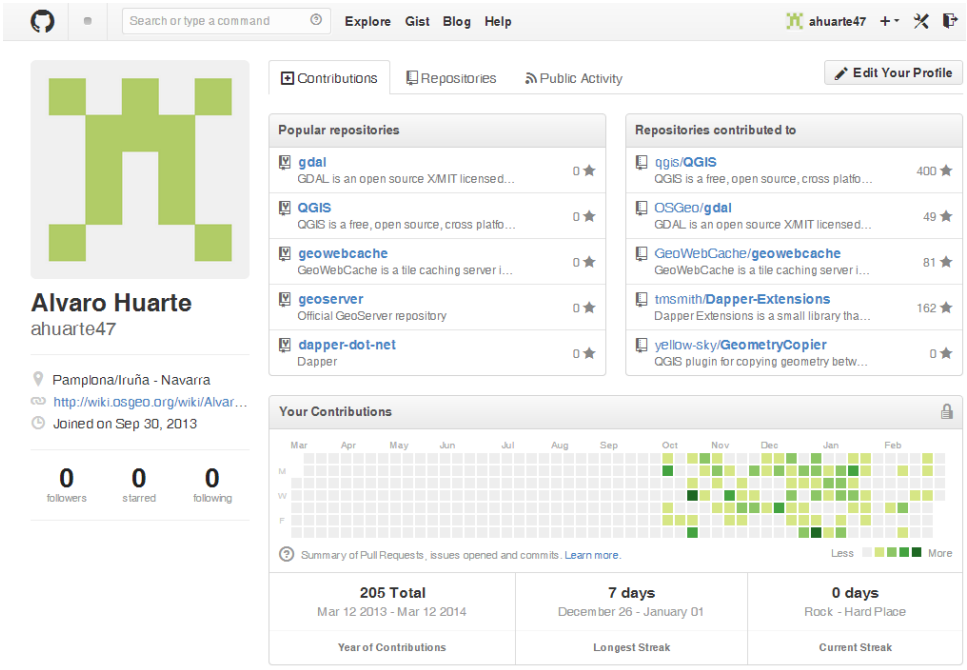

Figura 4: Página de -mi- usuario en github.

Nuestro siguiente paso es utilizar alguna herramienta que trate con el repositorio "github" del proyecto que nos interesa y que parte del hecho de tener que interaccionar con una dirección URL donde se alojan realmente los ficheros.

Para QGIS la URL del repositorio es:

https://github.com/qgis/QGIS.git

Si pretendemos subir cambios al repositorio base de QGIS, debemos en primera instancia hacer un clonado o “fork” de éste en nuestro propio repositorio “github”. Para ello hay que seguir unos sencillos pasos:

https://help.github.com/articles/fork-a-repo

Para nuestra comodidad hay un estupendo cliente desktop gratuito para Windows que nos permite la gestión cómoda de estos repositorios.

http://windows.github.com

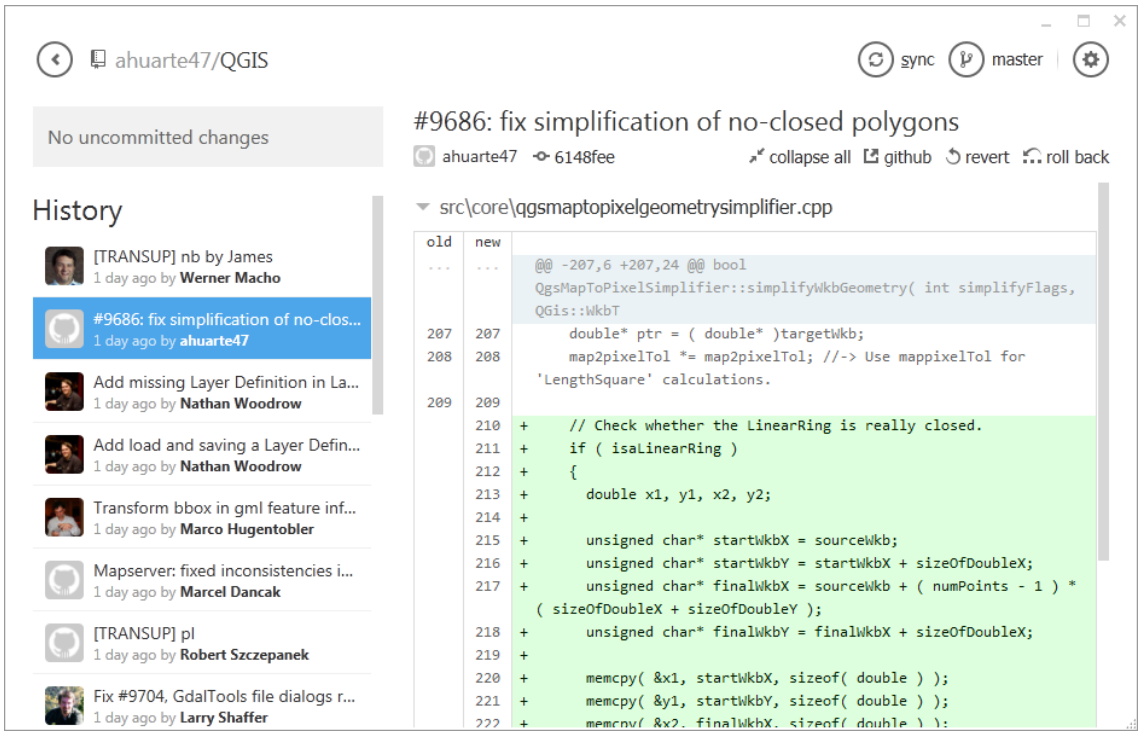

Figura 5: Cliente github para Windows funcionando.

Con este cliente podemos bajarnos a nuestra máquina la versión que queramos del código fuente, ver en detalle el histórico de cambios, quiénes los han realizado y finalmente, subir nosotros mismos cambios.

QGIS se desarrolla en una versión principal llamada “master” que es la que se entiende como la actual en desarrollo. Cuando el proyecto libera una nueva versión de la aplicación, se le da un nombre-número y se hace un clonado del estado actual de la versión “master” con esa nueva etiqueta definida. Es por ello que nos encontramos en github un listado de versiones etiquetadas estilo “release_x.x”, estas se corresponden al estado del código el día que se liberaron. A partir de ese momento quedan congeladas y desvinculadas de la rama “master” a menos que se decida incluir correcciones a errores graves encontrados en la aplicación.

No es propósito del artículo describir en detalle el trabajo con “github”, podremos consultar el gran abanico de documentos que hay Internet sobre su funcionamiento y gestión de los repositorios.

Compilar QGIS desde el código fuente

Una vez hemos bajado el código fuente para la versión o rama “github” que nos interesa, debemos compilar dicho código con la plataforma con la que estemos habituados a desarrollar. En este link podemos encontrar los pasos necesarios para los compiladores soportados.

https://raw.github.com/qgis/QGIS/master/INSTALL

En mi caso estoy usando “Visual Studio 9.0 C++ Express” para Windows [15], pero se pueden usar otros IDE’s de desarrollo, tanto para Windows como para los otros tipos de hardware soportados por QGIS. No dejo pasar este momento para recomendar el “Eclipse IDE” [16] por sus estupendas características y porque también es “opensource”, pero que cada uno elija el IDE con el que esté más cómodo.

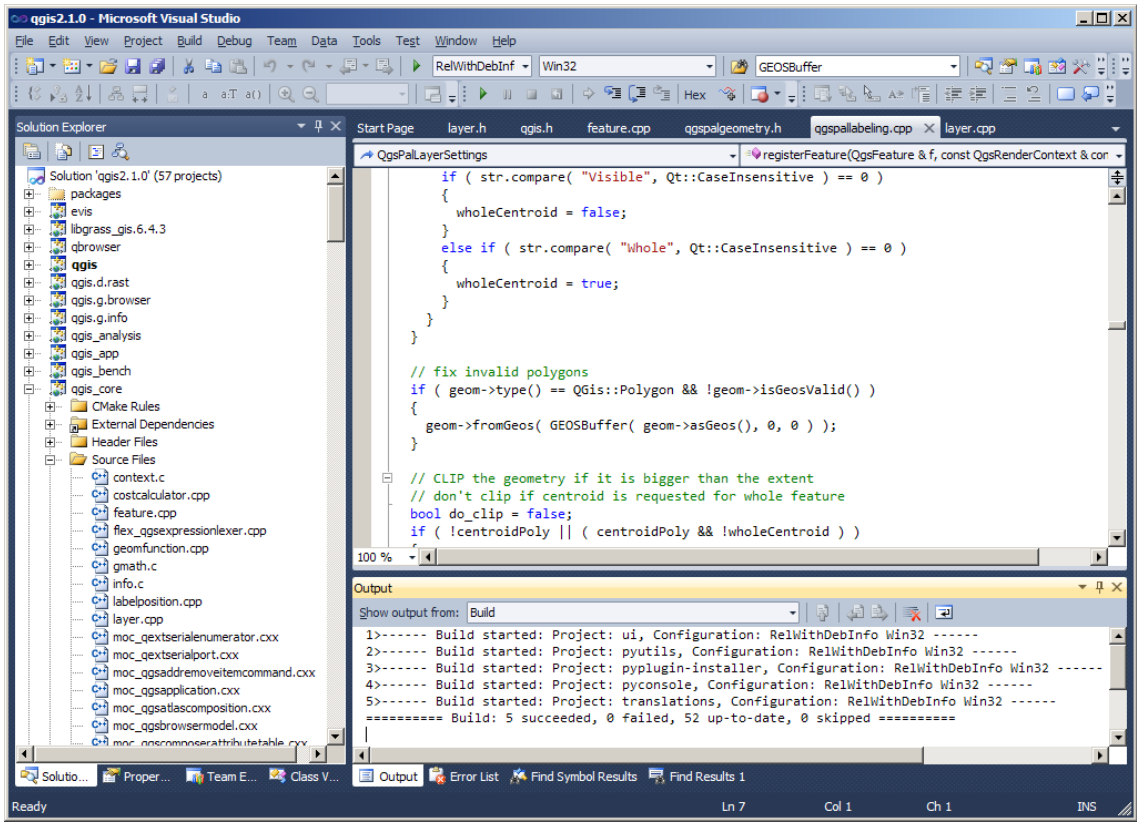

Figura 6: Visual Studio compilando QGIS.

Este punto es el más delicado, la compilación de cualquier software no sólo necesita del código directamente relacionado con él, sino de un conjunto de librerías externas que será necesario bajarse y asociar al proyecto principal. Para QGIS esto se describe en el anterior link para el IDE y arquitectura elegidos por nosotros.

No nos tenemos que preocupar por conocer todas las plataformas soportadas pues nosotros sólo compilaremos con nuestro IDE, editaremos el código cumpliendo unas normas de codificación detalladas en los recursos de desarrolladores, y subiremos sólo ese código editado a “github”. La compilación final del producto a las diferentes arquitecturas será a cuenta de los procesos montados a tal efecto por los administradores del proyecto.

Es importante recalcar que los usuarios recién llegados no pueden subir directamente cambios sino que sólo están habilitados para notificar un llamado “pull request” con nuestro nuevo código, sean correcciones de errores existentes o sean nuevas funcionalidades. Posteriormente los administradores principales del proyecto asignan uno o varios responsables de revisar, notificar errores o malas prácticas y/o sugerencias a nuestro trabajo para que al final sea éste “mergeado” a la rama principal del código fuente. Como es evidente, hay que ser muy cuidadoso con lo que se hace y en cómo se escribe el código, pero en definitiva no es otra cosa que seguir las normas de desarrollo detalladas en el proyecto.

Me gustaría recalcar en este punto la gran armonía y respeto que se percibe en el desarrollo de este proyecto, mi experiencia en GDAL-OGR y GWC también ha sido la misma. He sido un recién llegado y total desconocido y no he recibido otra cosa que apoyos, sugerencias bien intencionadas y cooperación positiva. Por eso, no nos debe dar miedo la colaboración en proyectos maduros, no hay gurús pedantes que hagan y deshagan, sino un equipo heterogéneo e internacional con multitud de puntos de vista y abierto al apoyo.

Nuestros “pull request” son acumulados en el repositorio principal de QGIS:

https://github.com/qgis/QGIS/pulls

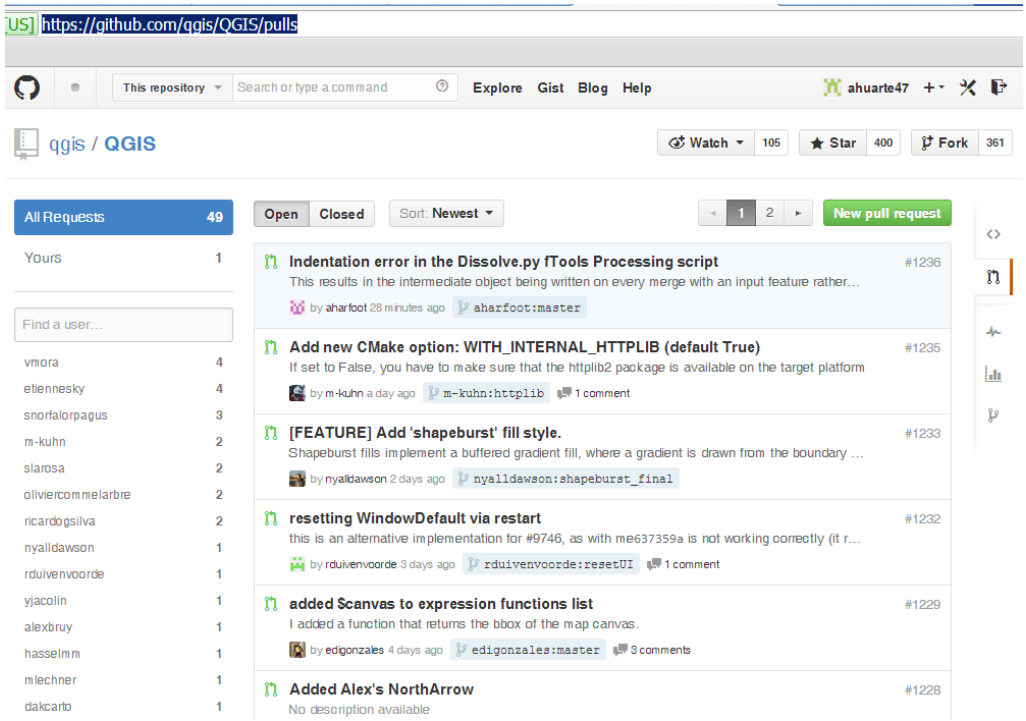

Figura 7: Lista de “pull request” pendientes.

Portal de bugs y nuevas funcionalidades

Relacionado con el desarrollo del proyecto se encuentra también disponible un portal donde se enumeran los “bugs” y nuevas funcionalidades que los usuarios notifican.

http://hub.qgis.org/projects/quantum-gis/issues

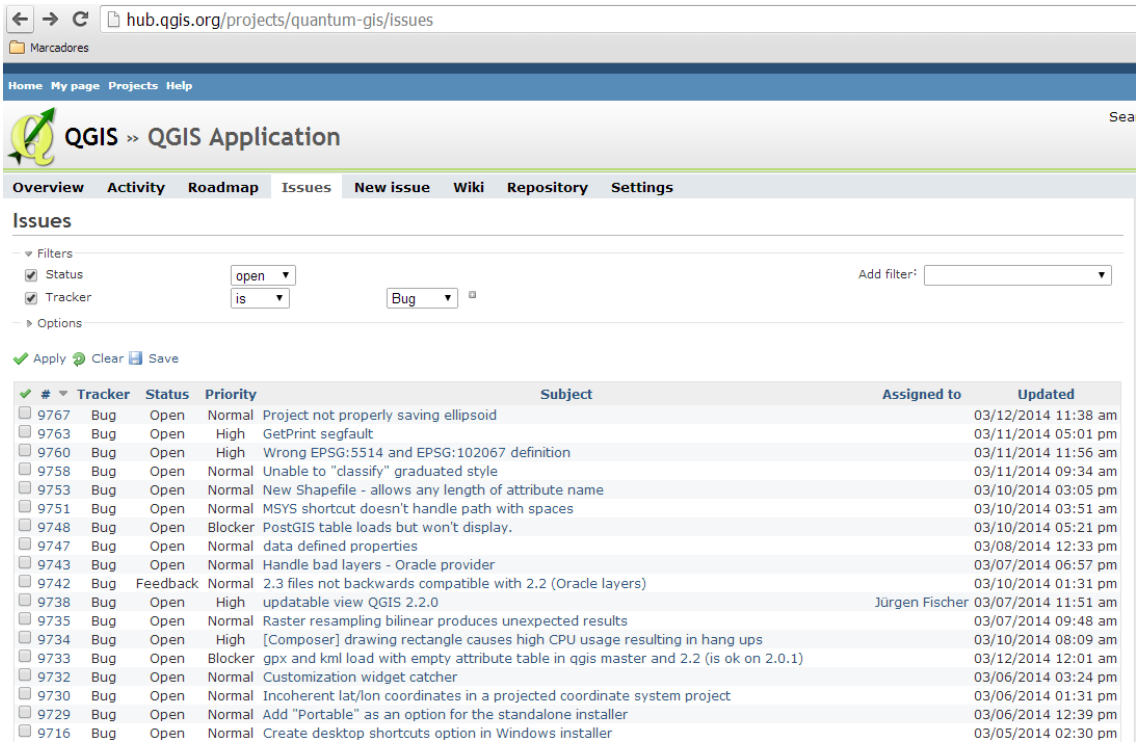

Figura 8: Lista de Bugs y nuevas funcionalidades pendientes.

En cualquier momento podemos añadir una nueva entrada a la lista, sea error o nueva petición de funcionalidad. Cuanto más se detalle la nueva entrada mejor, y si se trata de un error, incluir datos de prueba y la operativa para reproducirlo.

La resolución de errores suele llevarla a cabo alguno de los desarrolladores habituales del proyecto. En cambio, la implementación de una nueva funcionalidad debe ser admitida por su interés por alguno de ellos, o como es lógico, será necesario apadrinarla económicamente para que sea llevada a cabo. Vuelvo a comentar, como dije en un principio, que esto por muy “opensource” que sea, es implementado por alguien, con nombre y apellidos y, para nuevas peticiones no obvias, no esperemos resultados para el día siguiente.

Lista de correo para desarrolladores

El proyecto pone a nuestra disposición varias listas de correo agrupadas por roles, hay para desarrolladores, documentadores, estilos y conceptos de usabilidad, y de cuestiones de usuario en general.

Es necesario darse de alta para poder emitir correos a estas listas, el punto de entrada para desarrolladores es:

http://lists.osgeo.org/mailman/listinfo/qgis-developer

Figura 9: Histórico de correos relacionados con el desarrollo.

Vinculándonos a estas listas de correo podremos no sólo preguntar cuestiones sino además y, más importante aún, percibir el progreso del producto, valorar cómo se va “cociendo” la nueva versión o la resolución de errores, o el debate de las nuevas propuestas.

COMMITS PUBLICADOS

Como casos de “éxito” paso a describir algunos de mis desarrollos publicados. Estos “commits” implementan funcionalidades muy diversas; unos para soportar nuevas capacidades (o para mejorar funcionalidades ya existentes), otros para corregir errores reportados en las aplicaciones.

La lista de desarrollos los he agrupado por cada uno de los tres proyectos principales en los que he colaborado; GWC, QGIS y GDAL-OGR. Me gustaría resaltar que los cambios propuestos para los componentes GDAL-OGR fueron consecuencia directa de la mejora de funcionalidad detectada para los desarrollos de QGIS y evidencian también la necesidad de integración entre distintos paquetes de software, distintas comunidades de desarrollo, etc.

Mi experiencia ha sido más profunda en el desarrollo de QGIS y añado un link de acceso rápido a los “commits” de QGIS que los resume mejor.

https://github.com/qgis/QGIS/commits?author=ahuarte47

GeoWebCache (GWC)

Nuestra aportación a GWC fue de la mano de un proyecto específico en el marco del proyecto SITNA (Sistema de Información Territorial de Navarra) y a través de su IDE, IDENA. Este proyecto establecía la necesidad de publicar un nuevo servicio teselado de mapas (WMTS) de tipo RESTful.

El nuevo desarrollo posibilita generar en GeoWebCache (integrado o no en GeoServer) una caché en formato RESTful y que el producto base no soportaba en el momento de ejecución del proyecto. Añade además la posibilidad de establecer un directorio de salida opcional para evitar la costosa necesidad de trasiego de millones de ficheros típica de las caches (desde la carpeta de resultados a la carpeta de publicación final).

Evidentemente la implementación de la nueva funcionalidad supuso la modificación del código existente. El desarrollo se divide en dos partes diferenciadas:

1. Implementar la posibilidad de que el usuario pueda indicar una carpeta de salida de la caché distinta de la definida por defecto para GWC. Con ello conseguimos dos hitos, no volcar el resultado en la carpeta que GWC entiende como suya para usarla como su caché y el poder paralelizar en diferentes máquinas la generación de la caché a una misma carpeta destino evitando el trasiego final de ficheros para juntar el resultado de cada una de las salidas.

   La implementación de esta funcionalidad engloba tanto la modificación del interfaz de usuario en el panel Web de administración de caché, como la modificación del código fuente de GWC. Este desarrollo se ha publicado en un “pull request” en el repositorio github del producto para su aprobación por el grupo de gestores del proyecto.

   https://github.com/GeoWebCache/geowebcache/pull/200

2. Implementar un nuevo formato de caché con el esquema de tiles “estilo” RESTful. De igual forma que el punto anterior, la implementación ha englobado la modificación del interfaz de usuario y del código fuente para soportarla.

   Para darle generalidad a la modificación del software, escogí añadir al panel de administración una lista de formatos de caché de salida integrando en éste el formato existente hasta ahora (GWC) y que es el actualmente consumido por GeoWebCache y el nuevo (RESTful) que implementa la generación de caché según las necesidades del proyecto.

   https://github.com/GeoWebCache/geowebcache/pull/201

Estas modificaciones se han publicado en el sitio del proyecto en github.

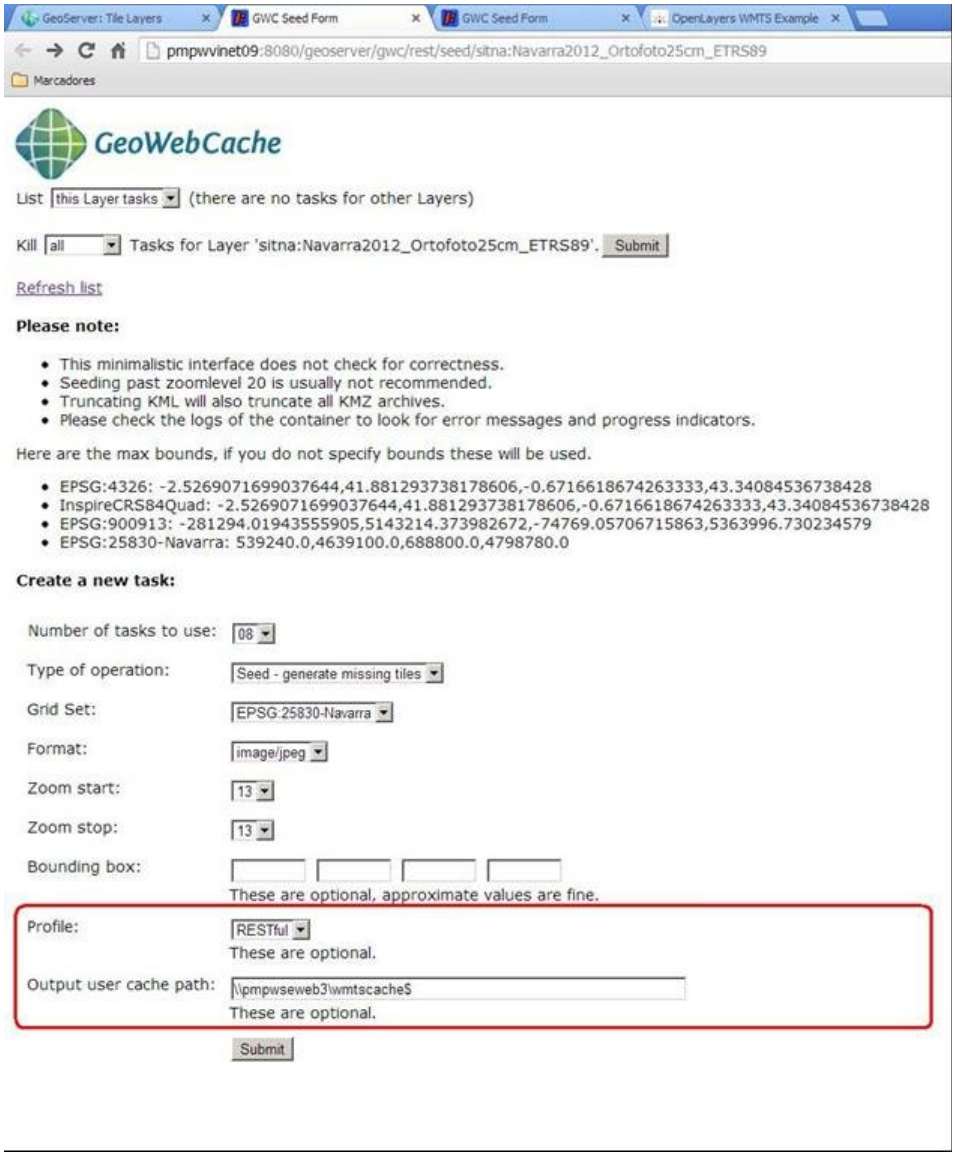

Figura 10: Modificación realizada en GeoWebCache.

QGIS

La lista de aportaciones es variada pero voy a detallar sólo algunas que considero interesantes por diferentes aspectos relevantes.

- *Set the canvas color in 'qgsprojectproperties::apply()' before refresh.*
  https://github.com/qgis/QGIS/commit/b2ad0a9bb914c67309b0e4c486b3493086b04a73

  Este simple "commit" corrige un error en QGIS por el que no se reflejaba correctamente en el mapa el cambio del "backcolor" configurado por el usuario de la aplicación. Lo considero interesante recalcar no por su funcionalidad, simple al extremo, sino por demostrar que todos podemos colaborar en lo que nos sintamos cómodos, implementando pequeños parches como éste, o codificando mayores cambios en el "core" si podemos y nos dejan ☺.

- *New keybindings, Del-key should delete feature...*
  https://github.com/qgis/QGIS/pull/1010
  http://changelog.linfiniti.com/qgis/version/21/#71

  Este "commit" fue implementado como respuesta a una nueva capacidad solicitada en la lista de peticiones de QGIS (http://hub.qgis.org/issues/9094). En principio sólo demandaba que la tecla "supr" eliminara las geometrías actualmente seleccionadas en la aplicación pero desembocó, gracias a la aportación de ideas de otros usuarios, en una revisión de varios "shortcuts" de QGIS para darle mayor coherencia al comportamiento de la aplicación y más ágil respuesta a tareas repetitivas en la edición de las capas. Esa colaboración de ideas y opiniones de diferentes usuarios y desarrolladores es lo que me gustaría resaltar de esta funcionalidad.

- *On the fly feature generalisation, fast rendering of features...*
  http://changelog.linfiniti.com/qgis/21/entry/on-fly-feature-generalisation/
  https://github.com/qgis/QGIS/pull/980
  https://github.com/qgis/QGIS/pull/1053

  Layer -> properties -> Rendering:

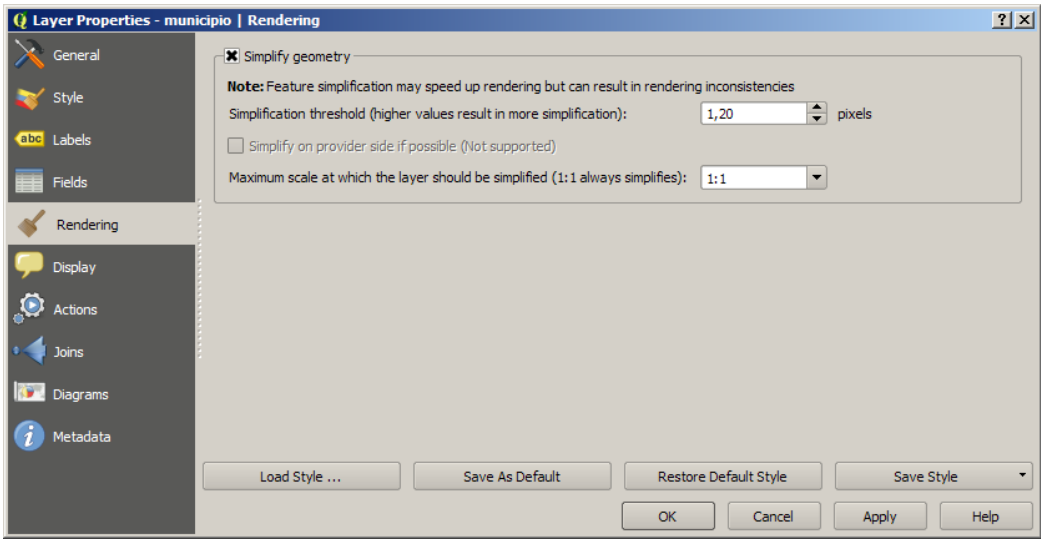

Figura 11: Configuración de la simplificación al vuelo de geometrías para el pintado rápido en QGIS.

Este “commit” fue iniciado por mí mismo como respuesta al estudio personal del comportamiento de QQIS cuando renderizaba vectores en versiones anteriores a la 2.2. QGIS, en comparación a otras aplicaciones equivalentes, tenía velocidades de pintado de geometrías con rendimientos tres o incluso cuatro veces inferiores para algunas capas vectoriales estudiadas.

El desarrollo al final implementó un conjunto de componentes que modifican el comportamiento base del pintado de la aplicación simplificando las geometrías a pintar utilizando diferentes técnicas. QGIS dispone ahora de una nueva configuración a nivel de capa vectorial que permite generalizar las entidades a pintar, y con ello una menor cantidad de puntos o vértices a pintar, sin pérdida perceptible de calidad visual. Esta simplificación, incluso la sustitución completa de la geometría por su BBOX (BoundingBox), se aplica a las entidades recuperadas desde los proveedores de datos, o se ejecuta directamente a nivel de proveedor como así lo soporta el driver de acceso a datos “postgis”.

Los tiempos de pintado son ahora comparables a los otros aplicativos equivalentes superando incluso a algunos de ellos, y se consiguen mejoras de rendimiento de hasta un 300% para capas vectoriales muy densas o voluminosas.

Lo importante de este “commit”, aparte de la funcionalidad obtenida, es demostrar que también es posible colaborar en “profundos” cambios en las aplicaciones. Bien es cierto que hay que tener muy presentes las cautelas, opiniones y por supuesto certeros consejos de los más veteranos del proyecto, más aún si los cambios provienen de un recién llegado al proyecto. Hay una gran lista de desarrolladores y testeadores que me han guiado en la consecución de estos “commits” y que no puedo dejar de agradecer (Ver agradecimientos).

GDAL-OGR

Como he comentado anteriormente los “commits” propuestos para este estupendo paquete de componentes fueron consecuencia de la necesidad de mejora de funcionalidad para los desarrollos de QGIS. En concreto, para obtener un mejor rendimiento en el pintado de vectores en el mapa, propuse dos cambios para corregir dos cuellos de botella en el rendimiento y que no provenían finalmente del código en QGIS sino de los componentes que éste usa de las GDAL-OGR.

https://trac.osgeo.org/gdal/ticket/5272
https://trac.osgeo.org/gdal/ticket/5357

Los dos “commits” de las GDAL-OGR están relacionados con el proveedor de datos del formato shapefile cuando se usa en peticiones masivas de datos y críticas en rendimiento como son la visualización o renderización de vectores en un mapa. Con estos cambios ya aceptados en el producto, se consigue una mejora apreciable en la carga de las geometrías que redunda en el rendimiento del pintado masivo de datos que un desktop como QGIS debe ejecutar.

CONCLUSIONES

Desde Tracasa (www.tracasa.es) la ejecución de estos hitos nos ha servido para conocer y valorar de primera mano la plena y satisfactoria integración del software "opensource", que cubre en sí mismo un gran abanico de funcionalidades genéricas que los proyectos necesitan, así como responder a necesidades determinadas de proyectos concretos mediante el desarrollo de funcionalidades no implementadas en el software base.

Tracasa siempre ha tenido como objetivo principal la prestación de servicios basados en el uso de la información territorial, tanto para administraciones públicas como para organizaciones privadas. Desde esta perspectiva, tanto la empresa como sus técnicos, entre los que yo mismo me encuentro, percibimos los efectos beneficiosos en el desarrollo, fomento y distribución de software libre, que redundan en la "democratización" de la tecnología y en el interés público del conocimiento.

En este entramado de personas y empresas que rodean a los proyectos "opensource" se aglutinan los intereses de todos. Eso es bueno, sobre todo para el usuario final que se beneficia del resultado (y que a veces debe colaborar con el esfuerzo en tiempo o económico en una parte del desarrollo del producto), para los desarrolladores o empresas impulsoras que obtendrán beneficios tangibles desde diferentes conceptos (soporte, personalización, implantación, marketing, …) o simplemente por motivos desinteresados para la consecución de un bien tecnológico público y común.

AGRADECIMIENTOS

A Tracasa (www.tracasa.es) por ofrecerme la posibilidad de realizar esta comunicación y a mis compañeros por los sabios consejos en la elaboración de esta presentación.

A los desarrolladores y usuarios de los proyectos QGIS y GDAL-OGR que me han aconsejado en los desarrollos en que he colaborado de dichos productos:

wonder-sk - https://github.com/wonder-sk,
m-kuhn - https://github.com/m-kuhn,
nyalldawnson - https://github.com/nyalldawson,
gioman - https://github.com/gioman,
jef-n - https://github.com/jef-n,
nathanw2 - https://github.com/nathanw2,
rouault - https://github.com/rouault,
etc etc etc

REFERENCIAS

[1] Wikipédia: http://es.wikipedia.org/wiki/C%C3%B3digo_abierto
[2] QGIS: http://www.qgis.org/es/site/
[3] gvSIG: http://www.gvsig.com/
[4] Grass GIS: http://grass.osgeo.org/
[5] uDIG: http://udig.refractions.net/
[6] GeoServer: http://geoserver.org/
[7] MapServer: http://mapserver.org/
[8] OpenLayers: http://openlayers.org/
[9] LeaLeft: http://leafletjs.com/
[10] GDAL-OGR: http://www.gdal.org/
[11] Boundless: http://boundlessgeo.com/
[12] Deegree: http://www.deegree.org/
[13] GeoWebCache (GWC): http://geowebcache.org/
[14] Github: https://github.com/
[15] Microsoft VS Express: http://www.visualstudio.com/es-es/products/visual-studio-express-vs
[16] Eclipse IDE: https://www.eclipse.org/