

**APLICACIÓN DE UNA DE GUIA DE RECOMENDACIONES AL PROYECTO DE
FUENTE ABIERTA ZATHURACODE**

ANDRES FELIPE BAQUERO MARULANDA



**UNIVERSIDAD DE SAN BUENAVENTURA
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS
SANTIAGO DE CALI
2012**

**APLICACIÓN DE UNA DE GUIA DE RECOMENDACIONES AL PROYECTO DE
FUENTE ABIERTA ZATHURACODE**

ANDRES FELIPE BAQUERO MARULANDA

Trabajo de grado para optar por el titulo de Ingeniero de Sistemas

Director:

Ingeniero Diego Armando Gómez Mosquera



**UNIVERSIDAD DE SAN BUENAVENTURA
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS
SANTIAGO DE CALI
2012**

NOTA DE ACEPTACION

Aprobado por el Comité de Grado en cumplimiento de todos los requisitos exigidos por la Universidad San Buenaventura para optar al título de Ingeniero de Sistemas.

Jurado

Jurado

Santiago de Cali, Noviembre de 2012

DEDICATORIA

Quiero dedicar este gran logro a mi familia que siempre me ha apoyado en este largo proceso, y que finalmente culmina en esta fecha. Agradezco especialmente a mi mamá por su gran apoyo, cariño, comprensión y paciencia, que me ha conducido por el buen camino durante toda mi vida.

También quiero dedicar este logro a mi Excelente Asesor de Proyecto y a los docentes del Programa, quienes se encargaron de transmitirme su conocimiento durante toda la carrera, su respeto y su exigencia, que han logrado mejorarme en todos los aspectos.

Y por ultimo quiero dedicar este triunfo, a todas las personas allegadas, amigos y compañeros, que siempre me han apoyado desinteresadamente y me han ofrecido su cariño.

AGRADECIMIENTOS

Quiero expresar mis grandes y sinceros agradecimientos a:

Toda mi familia y en especial a mi mamá por apoyarme siempre, brindarme su cariño y acompañarme en este largo proceso.

Mi Asesor de Proyecto, el ingeniero Diego Armando Gómez Mosquera, y a todos los demás docentes y directivos por transmitirme sus grandes conocimientos y respeto.

Mis compañeros y amigos por apoyarme siempre.

A todos ellos, Muchas gracias.

CONTENIDO

	Pág.
RESUMEN.....	10
ABSTRACT	10
INTRODUCCIÓN.....	11
1 PLANTEAMIENTO DEL PROBLEMA	12
2 OBJETIVOS	13
2.1 OBJETIVO GENERAL	13
2.2 OBJETIVOS ESPECÍFICOS.....	13
3 REFERENTE TEORICO	14
3.1 SOFTWARE LIBRE	14
3.2 FUENTE ABIERTA.....	15
3.3 DIFERENCIAS ENTRE SOFTWARE LIBRE Y FUENTE ABIERTA.....	16
3.4 COMUNIDADES DE PROYECTOS DE FUENTE ABIERTA.....	17
3.4.1 ECLIPSE	17
3.4.2 LINUX.....	18
3.4.3 MOZILLA.....	18
3.4.4 APACHE.....	18
3.4.5 OPEN SOURCE	19
3.4.6 PYTHON	19
3.4.7 WIKIMEDIA	19
3.4.8 GNOME.....	20
3.4.9 XIPH.ORG	20
3.4.10 KDE.....	20
3.5 PROYECTOS DE FUENTE ABIERTA.....	21
3.5.1 FIREFOX.....	21
3.5.2 LINUX.....	21
3.5.3 LIBREOFFICE	21
3.5.4 ANDROID.....	22
3.5.5 THUNDERBIRD	22
3.5.6 VLC	22

3.5.7	GIMP	22
3.5.8	CLAMWIN	23
3.5.9	KDE.....	23
3.5.10	POSTGRESQL.....	23
4	DEFINICION DEL TEMA	24
5	GUÍA DE RECOMENDACIONES	26
5.1	Escoger el nombre del proyecto	26
5.2	Declarar los objetivos	26
5.3	Declarar que el proyecto es libre	27
5.4	Lista de características y requerimientos	27
5.5	Estado del desarrollo	27
5.6	Descargas	28
5.6.1	Numeración de versión	28
5.6.2	Componentes del número de versión.	29
5.6.3	Empaquetado	30
5.7	Control de versiones y seguimiento de errores	31
5.7.1	Control de versiones.....	31
5.7.2	Vocabulario	32
5.7.3	Escoger un sistema de control de versiones.....	32
5.7.4	Utilizando el sistema control de versiones	33
5.7.5	Seguimiento de errores	34
5.7.6	Interacción con las listas de correo.....	35
5.7.7	Prefiltrado del gestor de fallos	35
5.8	Canales de comunicación.....	36
5.8.1	Manejando el crecimiento	36
5.9	Pautas de desarrollo.....	37
5.9.1	Tomar nota de todo	38
5.10	Documentación.....	38
5.10.1	Disponibilidad de la documentación.....	39
5.10.2	Documentación para desarrolladores	39
5.11	Ejemplos de salidas y capturas.....	40
5.12	Hosting enlatado.....	40
5.13	Escogiendo una licencia	41

5.13.1	Aplicar la licencia a nuestro software	41
5.14	Manera de comunicarse	41
5.14.1	Evitar discusiones privadas	42
5.14.2	Uso de archivos.....	42
5.14.3	Retirar la mala educación	43
5.14.4	Hacer revisiones visibles de código	43
5.15	Anunciar	44
5.15.1	Publicidad.....	44
5.15.2	Anunciar vulnerabilidades de seguridad	45
5.16	Listas de correo	46
5.16.1	Prevenir el Spam	46
5.16.2	Archivo	47
5.17	Sistemas de chat en tiempo real.....	48
5.17.1	Archivando las conversaciones del chat	48
5.18	Wikis.....	48
5.19	Infraestructura social y política.....	49
5.19.1	Dictadores benevolentes	49
5.19.2	Democracia basada en el consenso	50
5.19.3	Si no hay consenso, vote.....	50
5.20	Dinero.....	51
5.20.1	Tipos de participación.....	51
5.20.2	Contratos indefinidos	52
5.20.3	Contrataciones	52
5.20.4	Documentación y usabilidad	53
5.21	Comunicaciones	53
5.21.1	Estructura y formato	54
5.21.2	Contenido	54
5.21.3	Tono	55
5.21.4	No enviar correos sin propósito	55
5.21.5	Temas productivos y temas improductivos	56
5.21.6	Gente difícil	56
5.22	Coordinando a los voluntarios	57
5.22.1	Conseguir el máximo de voluntarios	57

5.22.2	Delegar.....	57
5.22.3	Distinguir entre pedir y asignar	58
5.22.4	Supervisar después de delegar	58
5.22.5	Tratar cada usuario como posible voluntario	58
5.22.6	Gerente de parches.....	59
5.22.7	Gerente de traducción	59
5.22.8	Gerente de documentación.....	59
5.22.9	Gerente de errores	60
5.22.10	Gerente de preguntas frecuentes.....	60
5.22.11	Committers.....	60
6	APLICACIÓN DE LA GUIA DE RECOMENDACIONES.....	62
7	LISTA DE CHEQUEO.....	70
8	CONCLUSIONES	71
	REFERENCIAS BIBLIOGRÁFICAS.....	72

RESUMEN

Debido a la rápida evolución de la tecnología, que permite una comunicación desde cualquier parte del mundo y es mucho más fácil desarrollar un trabajo a larga distancia, cada vez más personas se muestran interesadas en iniciar un proyecto de fuente abierta o participar en alguno que ya exista. Partiendo de esta premisa, se hizo una investigación de este tipo de proyectos y se puede ver una gran cantidad, pero lamentablemente la mayoría de ellos no han tenido éxito, y esto puede deberse a una mala gestión por parte del director desde su inicio.

Basándose en esta realidad, se decidió aplicar una guía de recomendaciones al proyecto de fuente abierta Zathuracode, la cual contiene una serie de actividades y pasos que podrán ser ejecutados fácil y rápidamente por el director del proyecto conjuntamente con sus colaboradores a lo largo del ciclo de vida del proyecto.

ABSTRACT

Due to the rapid development of technology that allows communication from anywhere in the world and is much easier to develop a long-range work, more and more people are showing interest in starting a open source project or join one already there. On this premise, it was an investigation of this type of project and you can see a lot of these, but unfortunately most of them have not been successful, and this may be due to mismanagement by the director since its inception.

Based on this fact, it decided to implement a set of guidelines to Zathuracode open source project, which contains a series of activities and steps that can be easily and quickly implemented by the project manager in conjunction with its partners throughout the cycle project life.

INTRODUCCIÓN

A medida que va evolucionando la tecnología, al mismo tiempo va aumentando la necesidad de innovar el software que trae consigo. Por esta razón, las personas, sin importar si son desarrolladores o usuarios, se involucran cada vez más con los programas ya establecidos, participan durante el desarrollo de algunos otros que les interese o su objetivo es comenzar un proyecto desde cero partiendo de una idea propia.

Sin embargo, hoy en día, el panorama no es nada prometedor ya que la mayoría de proyectos de fuente abierta han fracasado, sencillamente porque se dejan a un lado y se deja de trabajar en ellos, de manera que no se sabe el momento en que se acaba. Existen otros proyectos que han sobrevivido pero son disfuncionales, es decir que funcionan pero no son amigables ni confiables.

Este documento ofrece una serie de técnicas que ayudan a prevenir errores comunes que pueden llevar proyectos al fracaso, colocando un paralelo entre lo que está bien hecho y lo que está mal hecho a la hora de encaminarse en un proyecto de software de fuente abierta.

Por otro lado, basándose en las recomendaciones, se ofrecen ejemplos para cada uno de los puntos mencionados y se muestra una implementación de los mismos.

Este documento está dirigido a desarrolladores y directores que estén pensando en iniciar un proyecto de fuente abierta, también a los que ya iniciaron uno y necesitan alguna ayuda, y finalmente a los que estén planeando involucrarse en uno que ya exista. No se necesita ser un experto, pero se deben tener conocimientos en código fuente, compiladores y parches.

1 PLANTEAMIENTO DEL PROBLEMA

En la actualidad, los desarrolladores de software, organizaciones y grupos de investigación entre otros están usando proyectos de fuente abierta, aunque no lo sepan. Estos proyectos nacieron con el inicio del internet y con la tecnología necesaria para compartir libremente información. Existen miles de proyectos de todo tipo, desde sistemas operativos, hasta programas de diseño grafico, navegadores, contabilidad, etc. [1].

En el portal sourceforge.net que agrega proyectos de fuente abierta, hasta el 2008 estaban registrados 172.000 proyectos, y se supone que este portal recoge menos del 10% mundial de todos los proyectos de fuente abierta. En base a esto se puede estimar que la cifra global era de 1,7 millones de soluciones libres iniciadas, y hasta esa fecha la población que usaba internet era de 1.700 millones aproximadamente en todo el mundo, lo que supone que habría un proyecto de fuente abierta por cada 1000 personas [2].

En 2011, este portal tenía registrados más de 280.000 proyectos en curso [1]. Esto demuestra que ha habido un rápido incremento en los últimos años, aproximadamente ha incrementado 62% en tres años.

Infortunadamente, la mayoría de estos proyectos fracasan, probablemente el porcentaje de proyectos que han fracasado es de 90 – 95%. Esto puede ser debido a que se deja de trabajar en el proyecto, también porque se entera que se está duplicando el trabajo de alguien más, o el proyecto no es placentero ni amigable, entre otras [3].

Para abordar un proyecto de fuente abierta, se deben tener en cuenta una serie de condiciones para llevarlo paso a paso al éxito. De ahí surge la idea de la investigación, la cual al finalizar, se espera contar con la implementación de cada una de las recomendaciones en el Proyecto de fuente abierta Zathuracode.

2 OBJETIVOS

2.1 OBJETIVO GENERAL

Aplicar una guía de recomendaciones al proyecto de fuente abierta Zathuracode que cumpla con criterios mínimos de calidad de un producto de software de fuente abierta.

2.2 OBJETIVOS ESPECÍFICOS

- Recopilar información referente a literatura, comunidades, industria y sitios de interés sobre proyectos de fuente abierta.
- Crear un manual guía con una serie de pasos a seguir para desarrollar un proyecto de fuente abierta y que cumpla con unos estándares mínimos.
- Proponer una lista de chequeo con las recomendaciones expuestas para llevar un control sobre el desarrollo de un proyecto.
- Realizar una prueba de concepto del manual guía al proyecto de fuente abierta Zathuracode.

3 REFERENTE TEORICO

3.1 SOFTWARE LIBRE

Es un movimiento [4] que nace con Richard Stallman a mediados de la década de los años 80', luego de que Stallman solicitara a un propietario de software, el código fuente de un programa de impresión, ya que había encontrado un error y necesitaba solucionarlo lo más rápido posible sin reclamar algo a cambio [5], y el dueño a pesar de la oferta se negó rotundamente a hacerlo.

De esta manera, nació el Software libre, en el cual Stallman propuso que los usuarios deberían tener 4 libertades principales:

- Libertad 0: la libertad de usar el programa, con cualquier propósito.
- Libertad 1: la libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.
- Libertad 2: la libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.
- Libertad 3: la libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie [6].

En pocas palabras, los usuarios tienen la libertad de copiar, distribuir, estudiar, modificar y mejorar el software [7].

Entonces, Stallman creó la Fundación para el software libre [8] y el Proyecto GNU [9], los cuales sirvieron de base para el movimiento de Software libre.

Este movimiento está basado en una filosofía y una ideología bien definida, la cual pone la libertad del software por encima de todo, más allá de aspectos técnicos.

Con todo lo anterior, se planteó una estrategia legal para desarrollar software libre y no convertirlo en propietario, e introdujo el término Copyleft [10], que es un método para hacer un programa libre, y todas las versiones modificadas y extendidas también sean libres. Es decir, que este método obtiene las 4 libertades mencionadas inicialmente.

3.2 FUENTE ABIERTA

El termino Fuente abierta aparece por primera vez en el año 1997 gracias a Bruce Perens [11], quien dio su descripción en un documento para la conferencia de desarrolladores Debian [12].

Luego, en el año 1998, se da paso a la Fundación del movimiento de fuente abierta gracias a John Maddog Hall, Larry Augustin, Eric Raymond, Bruce Perens, entre otros [13].

Este movimiento, al igual que el de Software libre, propone una serie de requisitos que se deben cumplir para que un programa se pueda catalogar de fuente abierta [14]:

1. **Libre redistribución:** el software debe poder ser regalado o vendido libremente.
2. **Código fuente:** el código fuente debe estar incluido u obtenerse libremente.
3. **Trabajos derivados:** la redistribución de modificaciones debe estar permitida.
4. **Integridad del código fuente del autor:** las licencias pueden requerir que las modificaciones sean redistribuidas sólo como parches.
5. **Sin discriminación de personas o grupos:** nadie puede dejarse fuera.
6. **Sin discriminación de áreas de iniciativa:** los usuarios comerciales no pueden ser excluidos.
7. **Distribución de la licencia:** deben aplicarse los mismos derechos a todo el que reciba el programa.
8. **La licencia no debe ser específica de un producto:** el programa no puede licenciarse solo como parte de una distribución mayor.
9. **La licencia no debe restringir otro software:** la licencia no puede obligar a que algún otro software que sea distribuido con el software abierto deba también ser de código abierto.
10. **La licencia debe ser tecnológicamente neutral:** no debe requerirse la aceptación de la licencia por medio de un acceso por clic de ratón o de otra forma específica del medio de soporte del software.

No se ha creado una licencia específica para plasmar la definición de Fuente abierta, sin embargo, el movimiento realiza certificaciones a las licencias que cumplan con los requisitos mencionados [4].

Después se estableció la Iniciativa de fuente abierta [15] como la organización administrativa del movimiento, que es una de las principales organizaciones defensoras de la comunidad Hacker, al lado de la FSF [8].

El movimiento de fuente abierta obtuvo la atención de las principales compañías de software consolidadas como Corel, Sun Microsystems e IBM [13].

3.3 DIFERENCIAS ENTRE SOFTWARE LIBRE Y FUENTE ABIERTA

Luego de todo lo mencionado anteriormente, donde se describió cada uno de los dos movimientos, se puede observar y deducir que comparten un pensamiento y un planteamiento en cuanto a la distribución del software, aunque existen algunas diferencias que pueden ser importantes para las empresas y los usuarios interesados y vinculados a estos movimientos.

Una de sus diferencias recae principalmente en que el Movimiento de fuente abierta es más flexible que el Movimiento de software libre, ya que sus seguidores no deben pensar en solo usar software de fuente abierta, sino que también pueden usar software propietario. También es más flexible a la hora de considerar que una licencia es de fuente abierta, ya que no requiere la existencia del término copyleft [10] y acepta licencias donde no se requiere que sus programas o versiones derivadas sean libres [4].

Otra de sus diferencias reside en que el objetivo del Movimiento de software libre es eliminar todas las restricciones hacia los usuarios y seguir el desarrollo compartiendo sus esfuerzos, mientras que el objetivo del Movimiento de fuente abierta es compartir el código y permitir a los usuarios continuar desarrollando y adaptarlo a sus propias necesidades [16].

También difieren en el significado, ya que el significado de fuente abierta suprime la confusión de libre como gratis utilizada en el Movimiento de software libre y se centra en los temas tecnológicos, en la calidad del producto y en su economía [11].

Por último, cabe mencionar que estos movimientos, tanto el de software libre como el de fuente abierta no son opuestos, únicamente se debe entender que son movimientos diferentes, que coinciden en cierta parte de su objetivo y que solamente difieren en algunos principios básicos [4].

3.4 COMUNIDADES DE PROYECTOS DE FUENTE ABIERTA

A medida que ha pasado el tiempo, el termino fuente abierta ha ido creciendo cada vez mas y se ha ido familiarizando con el software, esto quiere decir que las personas y usuarios, no necesariamente implicados, han manejado y han necesitado investigar sobre este, ya sea por interés de vincularse a él, o por conocerlo ya que de pronto en alguna ocasión necesitaran hacerlo. Gracias a esto, el Movimiento de fuente abierta ha tomado una gran importancia, tanto así, que reconocidas compañías se han interesado y se han vinculado a este movimiento.

A continuación, se describirán algunas de las diferentes comunidades de fuente abierta que existen y que han aportado para agrandar este movimiento, mencionando su definición y sus objetivos.

3.4.1 ECLIPSE

Eclipse es una comunidad de personas y organizaciones que colaboran en el comercio de usar software de fuente abierta. Sus proyectos están basados en la construcción de una plataforma de desarrollo abierta compuesta por marcos extensibles, herramientas y tiempos de ejecución para la construcción, implementación y administración de software a través del ciclo de vida. La fundación Eclipse es una organización sin ánimo de lucro, que sirve de apoyo para los Proyectos de Eclipse [17].

El proyecto Eclipse fue creado por IBM en noviembre de 2001 y apoyado por un consorcio de proveedores de software. Mientras que la Fundación Eclipse fue creada en enero de 2004 como una organización independiente para servir como administrador de la comunidad Eclipse.

La fundación Eclipse ofrece cuatro servicios a su comunidad, como lo son:

- Infraestructura de TI (Tecnologías de la información).
- Gestión de la PI (Propiedad intelectual).
- Proceso de desarrollo.
- Desarrollo de ecosistemas.

3.4.2 LINUX

La fundación Linux es un consorcio sin ánimo de lucro dedicado a fomentar el crecimiento de Linux. Se creó en el 2000, y patrocina el trabajo del creador de Linux, junto con empresas líderes en tecnología y desarrollo.

Esta fundación promueve y protege los avances de Linux, obteniendo sus recursos de los miembros y de la comunidad de desarrollo de código abierto para garantizar que Linux sea libre y avanzado [18].

La fundación Linux ofrece:

- Colaboración neutral y educación.
- Proteger y desarrollar el desarrollo de Linux.
- Establecer a Linux como una plataforma técnica.

3.4.3 MOZILLA

Es una organización sin ánimo de lucro dedicada a mantener el poder de la web en manos de las personas. Su comunidad está compuesta por usuarios, colaboradores y desarrolladores que trabajan para innovar en su nombre.

El proyecto Mozilla tiene una forma de trabajo basada en la comunidad con la que crea software de código abierto de primera calidad y desarrolla nuevas clases de actividades colaborativas [19].

Algunos de sus objetivos son:

- Definir una visión de internet.
- Poder comunicarse con la gente, sin importar si tiene conocimientos técnicos.
- Lograr que los colaboradores estén orgullosos de lo que están haciendo.
- Proporcionar un punto de partida para que otras personas mejoren su visión de internet.

3.4.4 APACHE

Es una fundación creada en 1999 como una membresía sin ánimo de lucro, que tiene como objetivo garantizar que los proyectos de Apache sigan existiendo más allá de los voluntarios individuales.

Esta fundación proporciona apoyo organizativo, legal y financiero para la comunidad de proyectos de software de código abierto, que ofrecen productos de software para el bien público. También proporciona un marco establecido para la propiedad intelectual y las contribuciones financieras, y mediante un proceso de desarrollo colaborativo, los proyectos se ofrecen a nivel empresarial y atraen a grandes comunidades de usuarios [20].

También cabe mencionar, que la licencia Apache facilita a los usuarios, comerciales e individuales implementar sus productos.

3.4.5 OPEN SOURCE

Es una corporación sin ánimo de lucro fundada en 1998 y que está libre de impuestos.

Esta corporación está formada para educar sobre los beneficios del software de fuente abierta. Una de las actividades más importantes que realiza es mantener la definición de fuente abierta para el bien de la comunidad [21].

Cabe mencionar, que esta corporación está involucrada en la comunidad de fuente abierta, y se puede destacar su capacidad, educación y defensa pública para promover el conocimiento y la importancia del software de fuente abierta.

3.4.6 PYTHON

Es una organización sin ánimo de lucro formada en 2001 que tiene los derechos de propiedad intelectual detrás del lenguaje de programación Python, manteniéndolo abierta para que todos lo puedan usar.

Esta organización tiene como objetivo proteger, promover y hacer avanzar su lenguaje de programación, para apoyar y facilitar el crecimiento de la comunidad internacional de desarrolladores de Python [22].

La comunidad global de Python se basa en el respeto mutuo, la tolerancia y el aliento para ayudar a los demás a la altura de estos principios.

3.4.7 WIKIMEDIA

Es una organización caritativa sin ánimo de lucro dedicada a fomentar el crecimiento, desarrollo y distribución de contenido de forma gratuita.

Ofrece algunos servicios en línea como enciclopedias, diccionarios, citas de texto, libros, textos, taxonomía, noticias, materiales de aprendizaje e imágenes [23].

3.4.8 GNOME

Es una organización sin ánimo de lucro que promueve los objetivos del proyecto Gnome, ayudando a crear una plataforma de software de computación gratis para el público.

Mientras los colaboradores se dedican a desarrollar, corregir errores y documentar los proyectos, la fundación se encarga de guiar en el proceso y proporciona los recursos y la infraestructura [24].

3.4.9 XIPH.ORG

Es una organización sin ánimo de lucro encargada de la producción de herramientas de dominio público para el manejo de archivos multimedia.

Su objetivo es apoyar el desarrollo de protocolos libres para promover los mercados de negocios.

La fundación también ha proporcionado protección a diferentes grupos de desarrollo de software libre relacionados con los formatos multimedia, de los cuales la mayoría tienen su propio grupo de desarrolladores [25].

3.4.10 KDE

Es una comunidad internacional que coopera en el desarrollo de software libre y de fuente abierta. Ha desarrollado una serie de aplicaciones que tienen como objetivo la comunicación, el trabajo, la educación y el entretenimiento.

Esta comunidad se enfoca en buscar soluciones innovadoras a problemas viejos y nuevos, permitiendo la experimentación [26].

3.5 PROYECTOS DE FUENTE ABIERTA

De la misma manera que existen fundaciones que trabajan y apoyan el movimiento del software de fuente abierta, también existen proyectos que son los encargados de impulsar y de incentivar a las personas para que se unan a este movimiento.

Existen proyectos de software que tratan de diferentes temas y pertenecen a diferentes ramas, por ejemplo sistemas operativos, navegadores, reproductores, editores de imágenes, antivirus, entre otros.

En este apartado, se describirán algunos de los diferentes proyectos de software de fuente abierta que existen, mencionando sus características.

3.5.1 FIREFOX

Mozilla Firefox es un navegador web libre y de fuente abierta desarrollado por la fundación Mozilla [27]. Es el segundo navegador más usado de internet. Está disponible para diferentes sistemas operativos como lo son Windows, Linux y Mac. Su primera versión fue publicada en el mes de noviembre del 2004 [28].

3.5.2 LINUX

Linux es un sistema operativo libre y de fuente abierta [29]. Se ha diseñado y desarrollado por una gran cantidad de programadores en todo el mundo y es coordinado por el director del proyecto, Linus Torvalds. La primera aparición de este sistema operativo fue en el año 1991 [30].

3.5.3 LIBREOFFICE

LibreOffice es un grupo de aplicaciones gratuito y de fuente abierta disponible para Windows, Linux y Mac. Entre sus aplicaciones se encuentra un procesador de textos, una hoja de cálculo, un generador de presentaciones multimedia, un editor de gráficos, un motor de base de datos y un editor de ecuaciones [31].

LibreOffice fue creada en octubre del 2010 como una bifurcación de la suite OpenOffice.org por temor a ser descontinuada por Oracle. Su primera versión con su nuevo nombre fue publicada en esa misma fecha [32].

3.5.4 ANDROID

Android es un sistema operativo móvil basado en Linux, y está enfocado para poder utilizarse en dispositivos móviles como por ejemplo teléfonos inteligentes, tabletas, entre otros. Está desarrollado de forma abierta, y se puede acceder al código fuente y al listado de incidencias donde se pueden ver los problemas no resueltos y se pueden agregar problemas nuevos [33].

Android cuenta con una gran comunidad de personas desarrollando aplicaciones para extender sus funcionalidades, las cuales pueden descargarse gratuitamente en su tienda oficial llamada Google play [34], aunque también pueden encontrarse en otras tiendas no oficiales para Android.

Su primera publicación se realizó en el mes de septiembre del 2008 [35].

3.5.5 THUNDERBIRD

Mozilla Thunderbird es un cliente de correo electrónico desarrollado por la Fundación Mozilla. Es multiplataforma, es decir, funciona en diferentes sistemas operativos y es de código abierto [36].

Thunderbird cuenta con un asistente de migración, un asistente de configuración de correos, libreta de direcciones, recordatorio, administrador de actividad, búsqueda y archivado de mensajes.

Su primera versión fue lanzada en el mes de julio del 2003 [37].

3.5.6 VLC

VLC Media Player es un reproductor multimedia libre y de fuente abierta. Reproduce archivos, discos, cámaras web y dispositivos. Es multiplataforma ya que se puede ejecutar en todas las plataformas incluyendo Windows, Linux, Mac, Unix [38].

El proyecto se inició en el año 1996 y su primera versión fue publicada en febrero del 2001 [39].

3.5.7 GIMP

GIMP es un programa de distribución libre que se utiliza para tareas como retoque fotográfico, creación y edición de imágenes. Puede ser utilizado como un simple

programa de dibujo, como un experto en retoque fotográfico, un convertidor de formatos de imagen, entre otros [40].

GIMP está diseñado para ser ampliado mediante plugins y extensiones, y así lograr añadir nuevas funcionalidades. Es multiplataforma y se presenta en varios idiomas.

Su primera versión fue lanzada en enero de 1996 [41].

3.5.8 CLAMWIN

ClamWin es un antivirus libre para Microsoft Windows 98/Me/2000/XP/2003/Vista. Provee una interfaz grafica de usuario, viene con su propio instalador y es de código abierto gratuito [42].

Presenta un alto porcentaje de detección de virus, un planificador de búsqueda de virus, actualización automática de la base de datos de los virus, entre otras.

Su primera versión fue publicada en noviembre del 2005 [43].

3.5.9 KDE

KDE es un proyecto de software libre encargado en la creación de un entorno de escritorio contemporáneo para estaciones de trabajo Unix, similar a los escritorios Mac OS y Windows. Sus aplicaciones están traducidas aproximadamente a 75 idiomas [44].

El proyecto se inicio en octubre de 1996, buscando crear una interfaz grafica para sistemas Unix.

Su primera versión fue publicada en julio de 1998 [45].

3.5.10 POSTGRESQL

PostgreSQL es un sistema gestor de base de datos relacional orientado a objetos y de fuente abierta, dirigido por un grupo de desarrolladores que trabajan de forma desinteresada y apoyados por organizaciones comerciales.

Algunas de sus características son: alta concurrencia, ofrece llaves foráneas, disparadores, vistas, integridad transaccional, herencia de tablas, tipos de datos y operaciones geométricas y soporte para transacciones distribuidas [46].

El proyecto fue iniciado en 1982 y su primera versión fue lanzada en el año 1989 [47].

4 DEFINICION DEL TEMA

En la actualidad, los desarrolladores de software, las organizaciones y los grupos de investigación están usando proyectos de fuente abierta, aun sin saberlo. Todos estos proyectos nacieron con el inicio del internet y con la tecnología necesaria para compartir libremente información. Existen miles de proyectos de todo tipo, desde sistemas operativos, hasta programas de diseño grafico, navegadores, contabilidad, etc. [1].

En el portal sourceforge.net que agrega proyectos de fuente abierta, hasta el 2011 tenía registrados más de 280.000 proyectos en curso [1], y se supone que este portal recoge menos del 10% mundial de todos los proyectos de fuente abierta. En base a esto se puede estimar que la cifra global era de 2,8 millones de proyectos iniciados [2].

Desafortunadamente, la mayoría de estos proyectos fracasan, probablemente el porcentaje de proyectos que han fracasado es de 90 – 95% [3]. Esto puede ser debido a diversos factores [3]:

- El proyecto llega a un punto donde deja de interesar.
- El proyecto no tuvo una buena administración ni tuvo buen orden al momento de desarrollarlo.
- No se establecieron claramente los objetivos del producto.
- Se va a tratar un proyecto, el cual ya ha sido desarrollado anteriormente.
- No se le hizo la publicidad necesaria para atraer colaboradores.
- Entre otras.

Para abordar un proyecto de fuente abierta, se deben tener en cuenta una serie de condiciones o aspectos para llevarlo paso a paso al éxito, entre ellos se encuentran [48]:

- Escoger el nombre del producto.
- Declarar sus objetivos.
- Características o requerimientos del proyecto.

- Descargas.
- Versiones.
- Canales de comunicación.
- Pautas de desarrollo.
- Documentación.
- Ejemplos de representar la información.
- Licencias.
- Manera de comunicarse.
- Anuncios.
- Lo que necesita un proyecto.
- Listas de correo.
- Vocabulario.
- Navegabilidad.
- Seguimiento de errores.
- Sistemas de chat.
- Wikis.
- Sitio web.
- Infraestructura.
- Dinero.
- Contratos.
- Publicidad.
- Obstáculos.
- Crecimiento.
- Paquetes.
- Desarrollo.
- Coordinando los voluntarios.
- Copyright.

Teniendo en cuenta los aspectos mencionados anteriormente, se puede deducir que un proyecto de fuente abierta va mas allá de una idea, e incluso de tener un amplio personal trabajando en el desarrollo del mismo. Por el contrario, si se decide hacer énfasis en cada uno de esos aspectos, se ofrece cubrir una amplia variedad de características que permite asegurar una buena administración del proyecto.

5 GUÍA DE RECOMENDACIONES

Tomando como referencia el libro “Producir software de código abierto” [48], se crea una guía de recomendaciones, donde se describen gran parte de sus apartados, los cuales se presentan a continuación:

5.1 Escoger el nombre del proyecto

El nombre del proyecto será lo primero que un usuario o un desarrollador se encontrara. Un buen nombre no hará que el proyecto tenga éxito inmediatamente y un mal nombre no hará que fracase, aunque en realidad puede evitar el interés del usuario, ya sea por no tomarlo en serio o por no recordarlo [49].

Las consideraciones que se deben tener en cuenta a la hora de escoger un buen nombre son:

- Se debe relacionar o dar alguna idea de lo que el proyecto hace, esto permitirá que se recuerde rápidamente.
- Sea fácil de recordar, sin importar en qué idioma se escriba, aunque se debe tener en cuenta que algunas personas no lo leerán de la misma manera.
- No tenga el mismo nombre de otro proyecto y no infrinja una marca comercial, para no entrar a disputar legalmente ni crear confusiones de identidad.
- Este disponible como un dominio .com, .net y .org. Escoger probablemente .org para convertirlo en el sitio oficial del proyecto. Los demás dominios debe reenviar allí, únicamente para evitar que otras personas creen confusiones de identidad. Esto ayudara a que los usuarios tengan que recordar solo una dirección URL.

5.2 Declarar los objetivos

Luego de ingresar al sitio del proyecto, inmediatamente los usuarios o desarrolladores buscan una descripción rápida o una declaración de objetivos para despertar su interés en aprender más o en el peor caso desecharlo. Esto debe estar ubicado en un lugar estratégico de la página principal, preferiblemente debajo del nombre del proyecto.

La descripción o los objetivos deben ser cortos, concisos y concretos. Debe señalar que se trata de un proyecto de fuente abierta y que ninguna corporación dominara el desarrollo. También debe indicar que está dirigido a usuarios en todo el mundo y les permitirá contribuir en el proyecto. Y también señalar las plataformas en las que se puede ejecutar. Todo esto se debe mencionar basándose en los conocimientos previos de los lectores [50].

5.3 Declarar que el proyecto es libre

Los usuarios y desarrolladores que sigan interesados después de leer los objetivos, necesitan estar seguros que el proyecto es de fuente abierta [51].

En la página principal, se debe indicar específicamente y sin ambigüedades que el proyecto es de fuente abierta. Preferiblemente hacer esta declaración justo debajo de los objetivos del proyecto y exponer la licencia exacta por la cual se distribuye la aplicación.

5.4 Lista de características y requerimientos

Es muy importante diligenciar un breve listado de características y requerimientos, en el cual se mencionará lo que puede soportar el software a desarrollar y el entorno necesario para ejecutar la aplicación. Se debe pensar en un resumen del proyecto a la hora de escribirlo. Es importante indicar las funcionalidades que tiene hasta el momento, y se deben señalar las que están en proceso y las que se tienen como objetivo [52].

Este listado ayudara a aclarar el alcance y las limitaciones del proyecto.

Los usuarios y desarrolladores se basarán en esta información para decidir si involucrarse con el software, ya sea usándolo como herramienta o trabajando en su desarrollo.

5.5 Estado del desarrollo

Se debe ofrecer una página que suministre la información acerca del estado del proyecto en cuanto a su desarrollo. Tratándose de proyectos nuevos, las personas se interesarán en saber la distancia entre las promesas del software y la realidad en la que se encuentra. Siendo proyectos maduros, desearán saber cada cuánto

es mantenido, cada cuánto lanzan sus versiones, la facilidad para reportar fallos, etc.

En esta página, se aconseja listar los objetivos a corto plazo y las necesidades para llamar la atención de desarrolladores expertos en un tema en particular. También se debe dar un pequeño resumen de las versiones anteriores incluyendo sus características [53].

Cabe mencionar, que el manejo de versiones para el proyecto puede utilizar los términos alfa y beta. El término alfa significa que el software es estable en su funcionamiento pero que aún presenta fallos, mientras que el término beta significa que los fallos reportados fueron solucionados pero que aún no ha sido probado lo suficiente, y su objetivo es convertirse en una versión oficial del software.

5.6 Descargas

Se recomienda que el software pueda ser descargado como código fuente en formatos estándares, y no como archivos binarios o ejecutables, los cuales presentan un alto grado de dificultad de acuerdo a su compilación porque será mayor la cantidad de programadores que se den por vencidos y decidan abandonar el proyecto [54].

El software se debe distribuir de la forma más conveniente, estándar y sencilla posible.

Las tareas aburridas pero con un alto beneficio se deben hacer al inicio ya que quitan un obstáculo de entrada a potenciales usuarios y desarrolladores.

En el momento de lanzar una versión descargable del proyecto, se debe asignar un número único, para que las personas puedan comparar sus características con otra versión.

5.6.1 Numeración de versión

Cuando se hace oficial el lanzamiento de una nueva versión, los usuarios pueden asegurar que el software trae consigo una serie de características, como son:

- Los errores encontrados anteriormente han sido corregidos.

- Se han añadido nuevas inconsistencias.
- Se han añadido nuevas funcionalidades.
- Las opciones de configuración y de instalación pueden haber cambiado.
- Se han agregado cambios que puedan ser incompatibles con versiones anteriores.

Por estas razones, los usuarios temen a los lanzamientos de nuevas versiones, sobre todo cuando ellos piensan que el software ejecuta perfectamente lo que necesitan. Por otro lado, los números de la versión deben indicar el orden de las versiones y podrían señalar la naturaleza de los cambios en el lanzamiento [55].

5.6.2 Componentes del número de versión.

Los números de versión se sitúan al lado del nombre del proyecto para cada versión respectivamente. Estos números, aunque están separados por puntos no son decimales, y puede ser considerado como un estándar ya que a lo largo del tiempo se ha ido estableciendo de esta manera. No existe un límite para la cantidad de grupos de dígitos sin puntos, aunque la mayoría de proyectos utilizan máximo tres o cuatro. También suele añadirse la etiqueta 'Alfa' o 'Beta', que significa que precede a una versión con el mismo número de versión pero sin esa calificación.

Existen otras etiquetas, como 'Estable', 'Inestable', 'Desarrollo' y 'Candidata a versión', aunque las más utilizadas son 'Alfa' y 'Beta' seguida por 'Candidata a versión', pero se debe tener en cuenta que 'Candidata a versión' va acompañada por un número de incremento.

La numeración de la versión es un sistema de tres o cuatro componentes, en la cual un componente es un grupo de dígitos sin puntos, y su orden de importancia va de izquierda a derecha, siendo el primer componente de la izquierda el más importante, luego el segundo, y así sucesivamente. El primer componente recibe el nombre de Número mayor, el segundo componente Número menor, el tercero Número micro o Número de parche, y el cuarto Número de programa.

De esta manera, significa que un incremento en el Número mayor indica que hubo cambios importantes en el sistema; un incremento en el Número menor indica cambios de menor importancia; un incremento en el Número micro indica que hubo cambios triviales; un incremento en el Número de programa se utiliza para controlar las diferencias entre sus versiones [56].

A pesar que existen diferentes convenciones de numeración de versiones, los cambios y diferencias son mínimos.

5.6.3 Empaquetado

Usualmente, la forma de distribución del software es como código fuente, sin importar que se ejecute en forma de código (PHP, Python) o que requiera previa compilación (Java, C++). Cuando se cuenta con el software compilado, la mayoría de usuarios lo instalan desde paquetes binarios precompilados. El objetivo del paquete de código fuente es definir de forma segura la versión. Cuando el proyecto distribuye un software, quiere decir que cuando se hayan compilado e instalado los archivos de código fuente, reproducirá y ejecutara el software.

Existe una forma estándar de distribuir las versiones de código, y aunque haya diferentes, su desviación a esta norma es muy pequeña [57].

- **Formato:** para sistemas operativos Unix, se recomienda utilizar el formato 'tar', comprimido en 'compress', 'gzip', 'bzip', 'bzip2'. Para sistemas operativos Windows, se recomienda utilizar el formato 'zip', el cual se comprimirá de igual manera.
- **Nombre y diseño:** el nombre del paquete que se distribuirá debe contener el nombre del software, el número de versión y los sufijos del formato usado. Cuando se abra el paquete debe quedar un directorio con el mismo nombre excepto los sufijos del formato. En primera instancia debe aparecer un archivo 'README', el cual contiene las características del software, algunas referencias hacia la página principal del proyecto y hacia el archivo 'INSTALL'. El archivo 'INSTALL' también debe aparecer junto con el anterior, y debe ofrecer las instrucciones para compilar e instalar el software con cada uno de los sistemas operativos soportados. También debe aparecer el archivo 'COPYING', el cual se encarga de definir la licencia. Por último, debe aparecer un archivo 'CHANGES', en el cual se exponen cada una de sus características añadidas a la versión actual; sin embargo, en este archivo también aparecen las características de las versiones anteriores ordenadas de forma cronológica inversa, siendo la versión actual la primera en ubicarse.
- **Paquetes binarios:** el lanzamiento formal del proyecto es un paquete de código fuente, aunque la mayoría de usuarios instalaran el software desde paquetes binarios descargados desde un sistema distribuidor de software, o tomado manualmente desde el sitio web del proyecto o de un tercero. En este caso, binario no significa compilado, solamente se entiende que

permite que un usuario lo instale en su ordenador de forma fácil. Se debe evitar ofrecer paquetes binarios por cada cambio realizado, ya sea por un error reportado o por una funcionalidad añadida, ya que esto generará confusiones en el seguimiento de errores porque los usuarios confundirán este paquete como el lanzamiento oficial. En el caso de lanzar paquetes binarios no oficiales, se debe indicar y poner una advertencia en la página web del proyecto, en el cual indique que no es un lanzamiento oficial y presenta diferentes características.

5.7 Control de versiones y seguimiento de errores

Los desarrolladores que desean buscar fallos y añadir nuevas mejoras necesitan acceso en tiempo real a los últimos cambios, es decir necesitan acceso a un sistema de control de versiones. Si no se tiene un control de versiones, debe ser un objetivo el cual se comunica desde el principio.

En cuanto al seguimiento de errores del proyecto, se debe tener en cuenta que el número de fallos registrados depende de tres cosas: la cantidad total de errores del programa, la cantidad de usuarios utilizándolo y la conveniencia con la cual esos usuarios registran nuevos fallos. Cabe mencionar, que un proyecto con una base de datos de fallos amplia y bien mantenida generará una buena impresión.

En cuanto a los proyectos que están empezando, se recomienda que se enfatice en la juventud del proyecto, ya que de esa manera los usuarios se darán cuenta que el proyecto tiene una buena proporción de entradas basándose en que muchos errores han sido registrados recientemente [58].

5.7.1 Control de versiones

Es un sistema que se encarga de controlar los cambios realizados en el código fuente, la documentación y la página web de un proyecto. Es esencial contar con alguien que haya manejado alguna vez un sistema como este, ya que la mayoría de usuarios y desarrolladores esperan que el proyecto tenga un control sobre sus cambios. Si no se cuenta con una persona encargada del sistema de control de versiones, es muy factible que los usuarios no tomen en serio el proyecto.

Las ventajas de contar con este sistema son: comunicación entre desarrolladores, manejo de versiones, administración de fallos, equilibrio entre el código y el esfuerzo de desarrollo, y autorización en los cambios a los desarrolladores. La

característica principal del sistema es la administración de los cambios, para esto se debe identificar cada cambio al proyecto indicando la fecha y el autor de la modificación, y por supuesto que esté disponible para todos los usuarios [59].

5.7.2 Vocabulario

A continuación se dará la definición de algunos términos clave para administrar un sistema de control de versiones, independientemente del sistema que se maneje [60]:

- **Commit:** realizar un cambio en el proyecto y almacenarlo en la base de datos del control de versiones.
- **Log message:** es un comentario que se añade a cada commit indicando su tipo y su propósito.
- **Update:** solicitar la actualización del proyecto, de modo que se incluyan los últimos cambios realizados por otros desarrolladores.
- **Repository:** es la base de datos donde se almacenan los cambios.
- **Checkout:** obtener una copia del proyecto desde el repositorio.
- **Copia funcional:** copia del proyecto desde la cual se realizan y se prueban los cambios, para luego realizar un commit.
- **Diff:** es la representación de un cambio en el código.
- **Etiqueta:** se utilizan para preservar las capturas importantes del proyecto.
- **Branch:** copia del proyecto que no permite que sus cambios afecten al proyecto original y viceversa.
- **Merge:** mover los cambios de un branch a otro.
- **Conflict:** cuando dos desarrolladores intentan realizar un cambio en la misma porción de código.
- **Lock:** bloquear un archivo o directorio mientras un desarrollador lo utiliza, para que otro no pueda modificarlo al mismo tiempo.

5.7.3 Escoger un sistema de control de versiones

Esta tarea puede tornarse engorrosa y demorada, aunque en algunas ocasiones el sitio de hospedaje del proyecto ya cuenta con un sistema de control de versiones. En el caso que no sea así, se recomienda investigar un poco y consultar con otros desarrolladores para escoger un sistema y mantenerlo durante todo el proyecto [61].

5.7.4 Utilizando el sistema control de versiones

Versiones de todo. Aparte del código, se deben tener versiones de las páginas web, documentación, preguntas frecuentes, notas de diseño, es decir todas las cosas que requieran cambios, todo esto en el mismo sitio conjuntamente con el código para que los usuarios solo tengan que aprender a desenvolverse en un sistema de cambios [62].

Navegabilidad. Se recomienda que el repositorio del proyecto siempre sea accesible desde internet, de modo que permita consultar cualquier versión y observar sus características [63].

Correos de cambios. En el momento que un desarrollador realice un commit al proyecto, se debe enviar un correo automáticamente a la lista de correo de los desarrolladores mostrando el usuario que hizo el cambio, y las características de los archivos y directorios que han cambiado [64]. Algunas características que debe tener este correo son:

- Incluir el diff en el correo, aunque si es muy extenso se puede ofrecer una URL en donde se pueda encontrar completo.
- En la cabecera debe tener 'Responder a', la cual estará direccionada a la lista de correo de los desarrolladores.

Branch para evitar cuellos de botella. Se debe fomentar el uso de branch en el proyecto por parte de los desarrolladores para lograr formar un gran conjunto colaborativo sobre el mismo software. De esta manera se pueden comunicar unos con otros y llegar a un acuerdo en cierta funcionalidad, para luego hacer un merge con el proyecto original [65].

Singularidad de la información. Es muy importante tener claro que nunca se debe enviar el mismo commit dos veces, es decir que se debe enviar una sola vez al sistema control de versiones. Si se necesita hacer merge con otro branch, se recomienda que se tome el proyecto directamente desde el original donde ya se han aplicado los cambios [66]. Todo esto es por evitar que haya confusiones en el código, porque cuando se hace un merge con un branch se están creando algunos registros en el message log diferentes al del proyecto original.

Autorizaciones. Los sistemas de control de versiones ofrecen la posibilidad de conceder ciertos permisos a diferentes usuarios. Esto ya depende de la confianza de los directores del proyecto, aunque es recomendable que haya cierto grado de libertad para los desarrolladores, pero que se vayan ganando poco a poco la confianza y la credibilidad para poder participar cada vez más. Se puede empezar

por darle libertad en una parte del proyecto y que tenga que pedir la autorización para enviar un commit en el sistema de control, y así sucesivamente [67]. De esta manera se creará un ambiente de confianza durante todo el proyecto y será más fácil su expansión.

5.7.5 Seguimiento de errores

El sistema de seguimiento de errores se utiliza para seguir las solicitudes de tareas que puedan tener diferentes estados de principio y fin, con estados opcionales intermedios, y que acumulan información durante su existencia. Se debe tener en cuenta que aunque la mayoría de registros sean fallos, también pueden incluirse requerimientos [68]. El ciclo de vida que comúnmente cumple cada entrada, puede describirse en 6 pasos principales:

1. Alguien, sea usuario o desarrollador, crea una entrada del fallo o del requerimiento, ofreciendo un resumen y una descripción del mismo.
2. Los demás involucrados leen la entrada y añaden comentarios.
3. Se reproduce el fallo por parte de otra persona vinculada al tema.
4. Se identifica la causa y se estima el esfuerzo requerido para reparar el fallo.
5. Se fija una fecha o se decide la versión en la cual debe estar reparado el fallo.
6. El fallo es reparado. Luego de esto se debe indicar al inicio de la conversación que el tema ha sido cerrado y resuelto.

Existen algunos cambios en este ciclo que pueden ser muy comunes a lo largo del proyecto, como lo son:

- El tema es cerrado luego de ser ingresado, ya que puede ser un malentendido por parte del usuario. Pero se debe estar atento, ya que si el mismo fallo se presenta una y otra vez, puede ser un problema de la aplicación que requiera ser rediseñado.
- El tema es cerrado luego de ser ingresado, porque es un duplicado de otro tema. Si se trata de un tema ya cerrado, se recomienda reabrir el original y cerrar el actual como duplicado de este.
- El tema es abierto luego de haber sido cerrado anteriormente. Esto se puede presentar porque el desarrollador no pudo reproducir el fallo o porque no ha sido probado de la misma manera como se describió inicialmente.

El ciclo de vida de una entrada puede tener ciertas variaciones, pero básicamente será el mismo. Algo importante que se debe tener en cuenta, es reaccionar y responder a las entradas a medida que vayan llegando, para que las personas sepan que han sido tenidas en cuenta y que su fallo ha sido reportado con éxito, y esto animara al usuario a seguir involucrado en el proyecto. La necesidad de reaccionar oportunamente implica: que el sistema de seguimiento de errores se mantenga conectado a las listas de correos, para que cuando se registre un cambio genere un correo electrónico automático dando su descripción; la necesidad de reaccionar oportunamente también implica que el formulario que se debe diligenciar para registrar una nueva entrada debe incluir el correo electrónico de la persona que lo diligencia, ya que de esta manera puede ser contactada para solicitar más información.

5.7.6 Interacción con las listas de correo

Se debe asegurar que el sistema de seguimiento de errores no se convierta en un foro de discusiones, ya que este funciona más como un archivador, organizador de hechos, y hace referencia a las discusiones de las listas de correo [69]. Además, estos sistemas no están diseñados para mantener discusiones, y como si fuera poco, no todas las personas están pendientes del sistema de seguimiento de errores así como si lo están de las listas de correo.

5.7.7 Prefiltrado del gestor de fallos

Para evitar que la base de datos de fallos colapse debido a demasiados casos duplicados o inválidos, inicialmente se debe ofrecer un aviso en la página del seguimiento de errores, en el cual se explique la forma para saber si se trata de un error, si ya existe o la forma para registrarlo efectivamente [70].

Las dos técnicas utilizadas para evitar que la base de datos de fallos no se llene de basura son:

1. Delegar personas para vigilar los fallos reportados y que los puedan cerrar como inválidos o duplicados. Estas personas deben tener conocimiento suficiente del proyecto, y se pueden agrupar por categorías o módulos de software. En caso que los usuarios se equivoquen de modulo reportando un fallo, el delegado se debe encargar de responder a que modulo pertenece.

2. Solicitarle al usuario que antes de reportar un fallo en el sistema de seguimiento de errores, lo describa primero en alguna de las listas de correo para que algún otro usuario responda confirmando que es un error, o en caso contrario que no lo es o que ya ha sido resuelto.

Cabe mencionar que a pesar que se tomen estas medidas, siempre aparecerán reportes inválidos y duplicados a lo largo del proyecto, por eso se debe aceptar que la limpieza de la base de datos de fallos es parte de la rutina de mantenimiento, y se debe hacer lo posible por conseguir colaboradores.

5.8 Canales de comunicación

En el sitio web del proyecto se deben suministrar direcciones de listas de correo, salas de chat y foros, donde los desarrolladores y autores del proyecto puedan ser contactados, para que de esta manera los visitantes interesados y vinculados con el software puedan ofrecer una retroalimentación a los programadores. Por otro lado, no es necesario que los desarrolladores respondan a todas las solicitudes o preguntas que se generen por parte de los usuarios, ya que muchos no terminaran uniéndose a los foros, pero sí estarán tranquilos con saber que se podrían unir cuando lo deseen.

Al inicio del proyecto no es necesario separar los foros de los usuarios de los foros de los desarrolladores, ya que se está empezando y es mejor tener a todos los involucrados en una sola parte. Sin importar la proporción que exista entre usuarios y desarrolladores del software, no se puede asumir que todos están interesados en modificar el código, pero si se puede asumir que están interesados en seguir el curso del proyecto [71].

5.8.1 Manejando el crecimiento

A medida que el proyecto va creciendo, es decir cuando se incremente el número de visitantes y la cantidad de información buscada por ellos, surgirán muchísimas dudas que serán expresadas en el sistema de comunicación, como por ejemplo las listas de correo, y eso podrá convertirse en un problema ya que este crecimiento de usuarios no será proporcional al crecimiento de personas encargadas de responder todos los interrogantes.

Al principio, el sistema de comunicación, ya sean las listas de correo, foros o las salas de chat, funcionan perfectamente con algunos miles de usuarios y cientos de

mensajes o comentarios al día, pero cuando incrementa tanto que el usuario lector pierda el interés, estas se convertirán en una carga para sus miembros [72]. El efecto que tendrá en los visitantes será negativo y dejarán de estar suscritos a las listas de correo, salas de chat y foros, ya que sentirán que no son tomados en cuenta o escuchados entre tantos visitantes.

Existen dos estrategias para ajustar la comunicación durante el crecimiento del proyecto, las cuales son:

- **Identificar grupos de usuarios en los foros para formar un foro nuevo más especializado.** Al iniciar el proyecto, todos los usuarios del proyecto pertenecen a un solo foro o lista de correo general, en la cual se discuten las ideas, el diseño y los problemas de codificación. Más adelante, se van identificando diferentes módulos, en los cuales se dividirán los foros por diferentes temáticas, de manera que los usuarios puedan seleccionar sus listas de interés y en los cuales hablan de un tema en común.
- **Asegurarse que existen fuentes de información disponibles, actualizadas, organizadas y fáciles de encontrar.** Esta estrategia es un proceso que perdura durante todo el ciclo de vida del proyecto y cuenta con muchos involucrados, que serán los encargados de tener una documentación completa y actualizada para permitir a los usuarios consultarla cuando sea necesario.

5.9 Pautas de desarrollo

Este es un módulo importante a tener en cuenta, ya que aquí se explica la interacción entre desarrolladores y usuarios, y se describe como se deben hacer las cosas para contribuir al proyecto.

Algunos elementos básicos a incluir son:

- Enlaces a los foros.
- Instrucciones para reportar fallos
- Instrucciones para enviar parches.
- Explicación acerca del proceso de desarrollo del proyecto.

Se recomienda que en el momento en que un desarrollador en particular presente una restricción sobre todos los cambios en un módulo del proyecto, lo comunique a los demás vinculados desde el principio para evitar disgustos y abandonos del proyecto [73].

5.9.1 Tomar nota de todo

Es muy importante que en el proyecto exista un documento en el cual se registren todos los acuerdos y arreglos a los que se han llegado desde sus inicios. Estos acuerdos deben estar basados en las discusiones en las listas de correos y foros, y únicamente deben presentar su descripción.

En este documento no se deben describir o solicitar cuestiones obvias para el manejo del proyecto, como por ejemplo, la educación para expresarse o pedir un código limpio sin errores. Un buen punto en el cual se puede basar para construir el documento, es en el apartado de preguntas frecuentes de los usuarios, ya que son interrogantes que deberían estar solucionadas a la hora de ingresar como usuario nuevo al proyecto. Otro aspecto importante que se debe exponer en el documento es describir claramente la jerarquía política del proyecto y mencionar los procedimientos para la sucesión o el reemplazo de poderes [74].

Por último, es necesario dejarle claro a todos los involucrados en el proyecto, que las reglas, condiciones o acuerdos están sujetos a cambios y a ser reconsiderados nuevamente, siempre y cuando existan usuarios en desacuerdo.

5.10 Documentación

Una de las cosas más importantes para tener en cuenta es la documentación, sin importar si es algo básico e incompleto. La documentación, generalmente, es la parte donde más fallan los proyectos, ya que casi siempre se retrasa su inicio debido a que nunca está terminada.

La parte de mayor interés de la documentación para un usuario inicial es lo más básico: configuración de la aplicación, introducción del funcionamiento y algunas guías de tareas comunes. Pero al mismo tiempo esto es lo que mejor dominan los que conocen el software y se encargan de escribir la documentación, siendo un punto negativo porque muchas veces no ven desde el mismo punto de vista del lector, obviando pasos importantes.

Para redactar la documentación se debe utilizar un formato simple y fácil de modificar como HTML, texto plano o XML, esto podría facilitar el trabajo de documentar a una persona recién vinculada al proyecto [75].

Algunos criterios mínimos que deben cumplirse para escribir la documentación son:

- Informar al lector el nivel técnico que debe tener.
- Describir claramente la configuración del programa y brindar una prueba para confirmar que todo funciona correctamente.
- Ofrecer un ejemplo estilo tutorial para alguna funcionalidad común. Con esto, el usuario estará seguro que el software funciona y empezara a probar nuevas funcionalidades.
- Señalar los módulos que hacen falta por documentar. De esta manera se puede tratar como un requerimiento para futuros ayudantes.

El ultimo criterio mencionado puede ser aplicado a todo el proyecto, ya que se informara a los interesados en vincularse las deficiencias que presenta el software, preparándolos para lo que se van a encontrar y no llevarse sorpresas. Esto dará una cierta tranquilidad y confianza porque demuestra que se tiene conocimiento de su progreso.

5.10.1 Disponibilidad de la documentación

La documentación debe estar disponible en dos sitios: directamente en el sitio web del proyecto para las personas que deseen leerla antes de descargarlo y en el paquete descargable de la aplicación indicando que los archivos descargados son necesarios y están completos para ejecutar el software [76].

En cuanto a la documentación en línea, se debe ofrecer de manera que toda la información aparezca en una sola página HTML, ya que permite realizar búsquedas de una manera rápida sin importar que no se recuerde la sección en donde está.

5.10.2 Documentación para desarrolladores

Esta es la documentación que ayuda a los programadores a entender el código para luego ser modificado. Al principio del proyecto, no es necesario contar con una documentación para desarrolladores ya que a medida que va pasando el tiempo irán surgiendo preguntas sobre el código que tarde o temprano se repetirán, y esto es lo que justamente conlleva a que se escriba esta documentación [77].

Si únicamente hay tiempo para escribir un apartado de documentación, sin duda debe ser para los usuarios, ya que de igual manera tendrá que ser utilizada por los desarrolladores recién vinculados al proyecto para familiarizarse con él.

5.11 Ejemplos de salidas y capturas

En el caso que la aplicación presente una interfaz grafica o produzca una salida grafica para el usuario en el momento de ejecutarla, se deben ofrecer algunos ejemplos en el sitio web. Se debe tener en cuenta que un pantallazo le asegura al usuario que el software funciona, y suele ser mucho más convincente que un párrafo descriptivo, sin importar que el programa tenga fallos, sea engorroso para instalar o que la documentación este incompleta.

Hay algunas otras cosas que se pueden tener en cuenta para ofrecer en el sitio web del proyecto, como lo son las noticias, enlaces, búsqueda, entre otras. Aunque no es necesario al inicio del proyecto, se puede declarar como un objetivo a largo plazo [78].

5.12 Hosting enlatado

En la actualidad, existen sitios que ofrecen hosting y su infraestructura gratuita, contando con un área web, control de versiones, gestor de errores, zona de descargas, salas de chat, entre otros. Estos detalles varían entre los sitios pero prácticamente ofrecen lo mismo. Estos sitios presentan dos ventajas:

- La capacidad y ancho de banda del servidor.
- Las herramientas ofrecidas ya están configuradas y se respaldan por si solas.

Su desventaja:

- Estar obligado a usar sus opciones y configuraciones.

También puede haber algunas otras desventajas importantes, como el diseño de la página web del proyecto, ya que se podrá modificar hasta cierto punto, pero de igual manera aparecerán avisos publicitarios del sitio de hospedaje. Si finalmente se decide utilizar un sitio web de hospedaje, se puede pensar más adelante en utilizar un servidor propio, de manera que se utilicen nombres de dominio personalizados para la pagina principal del proyecto, y enlazarlos con el sitio publico de hospedaje sin necesidad de modificar la dirección del proyecto [79].

5.13 Escogiendo una licencia

Cuando se habla de licencias, se deben entender las implicaciones legales que traen consigo. Existen muchas licencias libres que pueden llamar la atención, aunque no se nombrarán todas en este apartado, se describirán las más comunes y seguramente las de mayor uso [80].

- **Licencias “Haz lo que quieras”.** Son las licencias más comunes, ya que indican que no restringe la libre distribución del programa e indica que su código no tiene ninguna garantía [81]. Esta licencia se puede utilizar en el caso que se desee que el software sea utilizado conjuntamente con software propietario.
- **Licencia GPL.** Se puede decir que es la licencia más usada para software de fuente abierta hoy en día [82]. Esta licencia se puede utilizar en el caso que no se permita utilizar el software con el software propietario.

5.13.1 Aplicar la licencia a nuestro software

Luego de escoger la licencia necesitada, se debe proveer en el sitio web del proyecto. Para ser más específico, se debe indicar la licencia escogida para el software en la página principal del sitio, solamente colocando su nombre y ofreciendo un enlace a otra página, en la cual se encontrará descrita totalmente esa licencia [83].

En el proyecto, se debe incluir la licencia en un archivo llamado COPYING o LICENSE, y en cada archivo con código fuente se debe indicar la ruta dentro del proyecto o un enlace URL para encontrar la licencia completa.

5.14 Manera de comunicarse

Es muy importante que las conversaciones se mantengan en el contexto del proyecto o una de sus funcionalidades y no se salgan de lugar. También asegurar que estas sean productivas para el crecimiento del software.

Muchas veces, la estabilidad y el crecimiento del proyecto no viene de imponer reglas o políticas, sino de un conocimiento global y colectivo difícil de describir y que evoluciona con el paso del tiempo.

Muchísimas veces, por no decir siempre, las personas que pertenecen a un grupo unido con un objetivo en común buscan algunas normas o conductas que los destaquen como parte del grupo, las cuales serán impuestas por el resto de vida del proyecto [84].

5.14.1 Evitar discusiones privadas

Por lo general, al publicar el proyecto, sus fundadores se verán en la situación de solucionar ciertas incógnitas, las cuales conllevan decisiones importantes que deben tomar y que existen pocas personas capaces de resolverlas. Algunas de las desventajas que se presentan son: el retraso en las conversaciones por foros y correos públicos, la formación de consensos, la dificultad de comunicación con las personas que creen saberlo todo y las que no entienden por qué se debe solucionar un problema en particular, entre otras.

Se aconseja no tomar decisiones importantes de forma privada, ya que eso conllevaría a espantar a los voluntarios del proyecto. Por el contrario, se recomienda permitir las discusiones públicas, sin importar que sean lentas y complicadas, ya que servirá de beneficio más adelante [85]. Algunas de las ventajas que se presentan son: muchas personas estarán pendientes del tema que se trate aquí, sin necesidad de participar activamente en la conversación pero se informarán sobre la aplicación; estas discusiones te ayudarán a ganar fluidez en cuanto a vocabulario a la hora de hablar sobre temas técnicos de la aplicación con personas que no están vinculadas; estas conversaciones quedarán guardadas en un archivo público, para evitar futuras discusiones sobre el mismo tema.

Para finalizar, se debe considerar que al hacer públicas las discusiones, existe una probabilidad de que haya alguna persona que este siguiendo pasivamente una discusión y espere el momento preciso para dar un gran aporte nunca antes pensado por los fundadores del proyecto.

5.14.2 Uso de archivos

Frecuentemente, las conversaciones y discusiones del desarrollo de un proyecto de fuente abierta son archivadas públicamente y se pueden buscar [86]. Esto es una excelente característica, ya que permite a los desarrolladores hacer referencia a ellas en cualquier momento que lo requieran, por ejemplo para contestar una pregunta de un usuario; en este momento se recomienda que se haga referencia

ofreciendo la dirección donde se encuentre la conversación archivada, ya que esto hará entender a los usuarios que buscando un poco podrán encontrar las respuestas necesarias y no hará perder tiempo a los desarrolladores.

En el caso que algunas de estas conversaciones estén inconsistentes debido a que son antiguas y se han añadido o cambiado funcionalidades, se recomienda que se cree una nueva conversación con la descripción vigente y declarar la anterior como obsoleta.

5.14.3 Retirar la mala educación

Cuando se decide hacer público un proyecto y abrir foros en los cuales se van a resolver las diferentes dudas, tanto de los usuarios como de los desarrolladores, se debe tener en cuenta que para mantener una cierta armonía y no entrar en una guerra, no se puede admitir por ningún motivo, que se utilicen palabras que puedan herir susceptibilidades y que puedan llevar a un enfrentamiento con otra persona [87]. Aunque en las discusiones públicas es muy frecuente que se presenten este tipo de cosas, se debe actuar de inmediato y enviar un mensaje que recalque la importancia de mantener la armonía en las discusiones sin señalar a ninguna persona, sin importar que parezcan palabras de profesor de escuela. Claro está, que en ese mensaje no se debe pedir un reconocimiento o una disculpa de la persona ofensora, ya que tal vez se sentirá resentida. También es trascendental no dar tanta importancia a la discusión y retomar rápidamente el tema principal del que se estaba hablando.

5.14.4 Hacer revisiones visibles de código

Una de las mejores prácticas para fomentar una comunidad productiva en desarrollo es hacerles saber que se están revisando sus cambios en el código y que todas las personas pueden verlo. Para esto, se hace uso de los correos de cambios, que se utilizan para enviar un correo mostrando los cambios hechos en el código y sus diferencias [88]. Esto es muy importante, ya que las personas se sentirán realmente vinculadas al proyecto, por eso se debe intentar decir algo sobre cada cambio registrado, ya sean críticas constructivas o elogios.

5.15 Anunciar

Cuando el proyecto este presentable, es decir, cuando por lo menos tenga una base solida a partir de documentación no necesariamente código funcional, el paso a seguir será anunciarlo al mundo. Se recomienda anunciarlo primeramente en freshmeat.net, que es un sitio encargado de anunciar noticias sobre nuevos proyectos. Luego, es importante anunciarlo en algunos foros o listas de correo donde el proyecto pueda generar algún interés en los participantes, y de esta manera se pueda atraer futuros usuarios y desarrolladores [89].

El anuncio en los foros y correos públicos debe ser corto y conciso, y mencionarlo una sola vez. En el mensaje se debe incluir: el asunto, el correo de contacto, la explicación de que se trata del lanzamiento del proyecto, el sitio web, las características y los requerimientos del proyecto.

Después de anunciar el proyecto, no se debe esperar una gran cantidad de visitas e interesados en participar, pero con el paso del tiempo se irá formando una comunidad de usuarios y de desarrolladores que comparten su código.

5.15.1 Publicidad

Generalmente, existen entre cuatro o cinco canales principales de distribución para llevar a cabo los anuncios simultáneamente:

1. En el caso que la pagina mas vista del proyecto sea la pagina principal, se recomienda colocar la propaganda ahí, la cual debe mostrar un resumen y un enlace a la noticia completa.
2. Se debe tener un modulo de noticias en el sitio web del proyecto, en el cual se registren todos los anuncios con su respectiva descripción detallada.
3. Si se cuenta con un actualizador RSS [90], se debe asegurar que el anuncio quede registrado.
4. Si se va a anunciar una nueva versión del software, se debe actualizar la entrada en donde se haya registrado el proyecto inicialmente, por ejemplo en Freshmeat.net.
5. Se debe enviar un correo a la lista de correo de anuncios, en la cual se indica en su definición que se trata de una lista de correo con poco tráfico.

Se debe tratar actualizar la información en todos los sitios donde este registrado el proyecto, para lograr una total consistencia [91]. Y es muy importante que se

actualice la información solamente cuando el cambio sea importante, por ejemplo cuando se vaya a lanzar una nueva versión.

5.15.2 Anunciar vulnerabilidades de seguridad

Los problemas de seguridad que existan se deben anunciar a todas las personas, incluso cuando se realiza un commit en el proyecto [92]. Por otra parte, la mayoría de proyectos de fuente abierta se han basado en estos criterios para anunciar un problema de seguridad:

- No se debe hablar sobre el problema de seguridad hasta que exista una solución, es decir que cuando se anuncie el problema se debe ofrecer la solución.
- Cuando alguien externo informa sobre el problema, se debe buscar una solución lo más rápido posible, ya que mínimo esa persona podría sacar provecho de la vulnerabilidad.

Estos criterios conllevan a unos principios estandarizados, como son:

- **Recibir el informe.** Debe existir una lista de correo exclusivamente para recibir los informes de problemas de seguridad, no deben tener los archivos que se pueden leer públicamente, y solamente pueden pertenecer a ella los usuarios y desarrolladores de confianza [93]. Esta lista no debe estar protegida contra spam o correo no deseado, ya que se podrían perder informes importantes de errores de seguridad.
- **Desarrollar la solución silenciosamente.** Para desarrollar la solución, se deben tener en cuenta algunos aspectos: la seriedad del problema, la facilidad de explotar el problema y el usuario informador del problema [94]. Luego de evaluar estos aspectos, se debe empezar a desarrollar la solución previamente discutida con los desarrolladores de la lista de correo, y mantener al tanto al usuario que informo sobre el problema. Se recomienda que cuando la solución este lista, se espere hasta la fecha de lanzamiento de un nuevo parche para aplicarla, ya que se podrían revelar los problemas de seguridad que presenta el proyecto.
- **Pre notificar.** Enviar correos a los administradores antes de su publicación, el cual contiene el informe de la vulnerabilidad y su solución indicando que sea prudente con la información [95]. Se recomienda enviar correo de pre notificación individualmente a cada uno de los usuarios de la lista para evitar que conozcan la dirección de los otros usuarios con los mismos

problemas, y enviarlo con el correo mostrado ya que algunos usuarios tendrán protección anti spam y no lo recibirán.

- **Distribuir la solución públicamente.** Se recomienda anunciar la solución al problema con la misma prioridad de una nueva versión, actualizando las entradas en noticias del proyecto, en el sitio web donde este registrado, y enviar un correo a la lista de anuncios del proyecto [96].

5.16 Listas de correo

Las listas de correo pueden ser consideradas las más importantes para llevar a cabo la comunicación entre los usuarios del proyecto, y se debe tener en cuenta que si algún usuario pertenece a un foro aparte de las páginas web, debería pertenecer a las listas de correo [97].

Se recomienda no dirigir las listas de correo a mano, y por el contrario, es mejor conseguir un software para el manejo de las listas, ya que este brinda las siguientes características:

- **Suscripción a través de correos o basada en web.** Cuando un usuario se registra, se reenvía un correo con mensaje de bienvenida y con las instrucciones para interactuar con el software y para cancelar la suscripción.
- **Suscripción al modo de resúmenes o al modo de mensaje por mensaje.** Los usuarios pueden escoger entre recibir diariamente la actividad resumida del día en las listas o recibir cada mensaje enviado a la lista.
- **Características para la moderación.** Se revisan los mensajes antes que lleguen a la lista para prevenir que no sea spam.
- **Interfaz administrativa.** Permite al administrador de la lista, manejar las direcciones de correo, ya sea para eliminarlas o crear nuevas.
- **Archivo.** Todos los mensajes enviados son almacenados públicamente para que los usuarios puedan verlo cuando lo necesitan.

5.16.1 Prevenir el Spam

Anteriormente, este tema era insignificante, ya que los correos basura no se presentaban con frecuencia, o mejor dicho, se presentaban rara vez. En cambio, en el día de hoy, debe ser un tema de suma importancia ya que la lista de correo se puede ver atacada por demasiados correos basura [98].

La prevención del Spam se divide en dos categorías, filtrar los mensajes y ocultar las direcciones en los archivos.

Para la filtración de los mensajes, muchas aplicaciones ofrecen las siguientes técnicas:

- **Solo permitir mensajes de los suscriptores a la lista.** Sería bueno hasta cierto punto, ya que también debería permitirse mensajes de personas que estén suscritas.
- **Filtrar los mensajes utilizando un programa de filtro de Spam.** La mayoría permite esta opción de filtro automático; aunque ninguno sea perfecto, se reducirán enormemente los correos basura.
- **Moderación.** Se utiliza para enviar el correo al administrador de la lista para que lo revise y decida aprobarlo o rechazarlo.

Para ocultar las direcciones en los archivos, se recomienda reemplazar la '@' en las direcciones de los usuarios cuando se registren, aunque esto puede ser contraproducente en ciertas personas, ya que sería difícil descifrarlo a la hora de enviar un mensaje directo. Por esta razón, se aconseja que se ofrezca una opción para que el usuario decida si reemplazar su dirección o no.

5.16.2 Archivo

Para configurar un archivador, se debe considerar lo siguiente:

- **Actualización rápida.** Se debe intentar que se archive cada mensaje instantáneamente, o cada cierto tiempo, el cual debe ser el más corto posible.
- **Estabilidad referencial.** Quiere decir que cuando se publique la URL del archivo, esta deberá permanecer por siempre, para permitir que los usuarios la tengan presente y la puedan indexar. También se recomienda ofrecer la URL en la aplicación de las listas de correos, para que las personas puedan ubicar su lugar en el archivo sin necesidad de visitarlo.
- **RespalDOS (Backups).** Los datos en los archivos son muy importantes, por esta razón se debe conocer donde se almacenan los mensajes y saber restaurar las páginas del archivo cuando sea necesario.
- **Soporte de los hilos.** Desde cada mensaje debe ser posible ir al grupo de mensajes relacionados a este y debe tener su propia URL.
- **Búsquedas.** El archivo debe permitir realizar búsquedas y que el usuario pueda especificar entre asunto y cuerpo del mensaje [99].

5.17 Sistemas de chat en tiempo real

Estos sistemas permiten a los usuarios que puedan realizar preguntas para que sean respondidas inmediatamente. Se recomienda utilizar algún sitio que permita crear canales de chat y que proporcione el control necesario para administrarlo.

Lo primero que se debe escoger es el nombre, por lo general se debe utilizar el mismo nombre del proyecto, en caso contrario el más parecido y fácil de recordar. Se debe publicar la dirección del canal en el sitio web del proyecto para que los usuarios puedan acceder fácilmente a él.

A medida que vaya creciendo el proyecto, se recomienda crear un canal para cada tema o modulo, y en este caso se debe indicar su tema en el tópico del canal [100].

5.17.1 Archivando las conversaciones del chat

No es necesario archivar todas las conversaciones del chat, ya que muchas veces son conversaciones informales, con poca gramática y pueden ser opiniones al aire, sin mencionar que son públicas. Si se decide archivar las conversaciones, se debe indicar de esa manera en el tópico del canal y proporcionar el enlace del archivo donde se almacenan [101].

5.18 Wikis

Los Wikis son sitios web que permiten a los usuarios editar y extender su contenido. Es una mezcla entre un sistema de chat en tiempo real y las páginas web, pero no trabajan en tiempo real, permitiendo a los usuarios considerar y pulir sus cambios, y aparte de esto es mucho más fácil de editar que una página web [102].

Para ofrecer un wiki, se debe tener claro la organización de las páginas y que sean visualmente atractivas, para que los usuarios sepan fácilmente la manera de contribuir. También es muy importante, hacer énfasis de estos estándares en el wiki para que cualquier visitante pueda dirigirse a buscar información. La mayoría de wikis presentan ciertos problemas, como son:

- **Falta de principios de navegación.** Cuando un sitio web está mal organizado, ya que los visitantes no saben la sección en la que están.
- **Información duplicada.** Cuando se tiene información similar en diferentes paginas del sitio web.
- **Audiencia objetivo inconsistente.** Cuando se crean nuevos contenidos por parte de demasiados autores.

La mejor solución y la más común, es tener estándares editoriales, publicarlos y aplicarlos, de modo que las demás personas empiecen a imitarlo, mediante una plantilla bien elaborada para seguir.

5.19 Infraestructura social y política

En este apartado se explicarán algunos interrogantes que aparecen al inicio de todos los proyectos de fuente abierta y libre, como lo son su funcionamiento, mantenimiento y la toma de decisiones.

Se mostrará la estructura para llevar el proyecto al éxito en cuanto a calidad técnica, salud operacional y capacidad de sobrevivencia. La salud operacional es la capacidad de introducir nuevo código, nuevos desarrolladores, y asumir la responsabilidad de los informes de errores ingresados. La capacidad de sobrevivencia es la capacidad que tiene el proyecto de existir sin depender de un usuario o desarrollador en particular. La calidad técnica se alcanza cuando se tiene una base de desarrollo bastante amplia y un fundamento social [103].

5.19.1 Dictadores benevolentes

Un dictador benevolente es la persona encargada de tomar las decisiones, las cuales se espera que sean tomadas sabiamente [104]. Esta persona es elegida por la comunidad del proyecto y no se debe comportar como amo. Por lo general, el dictador benevolente no toma las decisiones, ni siquiera la mayoría, por el contrario espera que los participantes lleguen a un acuerdo mediante el intercambio de ideas. Luego de esto, si no se puede llegar a un consenso y se desea que alguien tome la decisión, aparece el dictador benevolente para hacerse sentir y guiar la decisión como lo requiera.

Generalmente, los dictadores benevolentes de los proyectos de fuente abierta cumplen con algunas características, las cuales son:

- Tener cierta delicadeza para juzgar su influencia en el proyecto, de manera que no imponga su posición de líder y permita a los usuarios opinar abiertamente.
- No necesita tener una habilidad técnica superior a la de los otros usuarios y desarrolladores del proyecto, solamente necesita un conocimiento suficiente para trabajar en el código y el funcionamiento del software.
- Necesita tener experiencia y conocimiento general de diseño.
- Comúnmente, un dictador benevolente es el fundador del proyecto.

La decisión de tener o no dictadores benevolentes se toma al inicio al inicio del proyecto y se orienta todo en ese camino.

5.19.2 Democracia basada en el consenso

Con el paso del tiempo, el proyecto pasa del sistema de dictador benevolente al sistema democrático, y por lo general no vuelve atrás [105]. Existen algunos elementos comunes en el funcionamiento de estos sistemas, como son:

- El grupo funciona por consenso la mayoría del tiempo.
- Existen un mecanismo formal para realizar votaciones cuando no se logre un consenso.

Las discusiones que se presentan son generalmente sobre asuntos técnicos, forma correcta de solucionar un error, conveniencia de agregar un asunto, la forma de enlace de un documento, etc. Cuando se termina una conversación, alguien debe concluirla con un último mensaje, en el cual se brinde un resumen de lo discutido y se describa el acuerdo al que se ha llegado, para así dar una última oportunidad de oponerse a alguien que se encuentre en desacuerdo.

5.19.3 Si no hay consenso, vote

Cuando en una discusión no se puede llegar a ningún acuerdo, la solución es votar. En este caso, se escogen personas para que realicen una síntesis con los argumentos y expongan los puntos de acuerdo y desacuerdo [106]. Estas síntesis pueden ser usadas como propuestas electorales para los participantes.

Se recomienda que los votos sean abiertos, ya que se trata de asuntos discutidos públicamente, y que cada participante ponga su voto en la lista de correo del proyecto, de manera que quede archivado y se pueda verificar el resultado.

5.20 Dinero

En este apartado se verá la forma de conseguir fondos en un proyecto de fuente abierta, está dirigido a los desarrolladores y a los directores, que son los encargados de coordinarlos.

Anteriormente, los fondos corporativos provenían de los sueldos de los administradores que tenían un consorcio de un software que había sido puesto en línea por uno de ellos para atraer a los demás y conseguir contribuciones. Al día de hoy, las corporaciones tienen en cuenta el beneficio del software de fuente abierta y se involucran en su desarrollo. También se espera que lleguen donaciones y patrocinadores. En el caso de conseguir patrocinadores financieros dará más credibilidad al proyecto, ya que los usuarios lo tomarán más serio y tendrá confianza en él [107].

Sin embargo, se debe tener control sobre los fondos recaudados por el proyecto. Para esto, se recomienda seguir tomando las decisiones conjuntamente con todos los vinculados, para que de esta manera no sientan que las decisiones solo están disponibles para el mejor postor, y esto conllevaría a que terminen abandonando el proyecto.

5.20.1 Tipos de participación

Los proyectos de fuente abierta consiguen fondos por diversas razones, como son [108]:

- **Compartir la carga.** Cuando diferentes organizaciones se encuentran realizando el mismo trabajo pero separadas, cada una por su lado, escribiendo código similar para solucionar problemas parecidos.
- **Aumentar servicios.** Cuando una organización vende servicios de los cuales depende o vende programas de fuente abierta particulares.
- **Apoyar las ventas de hardware.** Esto cuenta para ambas partes, software y hardware, ya que si existe una buena cantidad de software de fuente abierta habrá un incremento en el valor de los ordenadores y viceversa.
- **Socavar la competencia.** Cuando algunas organizaciones patrocinan proyectos de fuente abierta para atacar los productos de la competencia, sin importar que sean propietario o de fuente abierta.
- **Mercadeo.** Estar asociado con un proyecto de fuente abierta generara muy buena publicidad.

- **Licencias duales.** Cuando se ofrece el software bajo una licencia que permite utilizarlo conjuntamente con un software propietario, y que pueda participar activamente en el desarrollo del mismo.
- **Donaciones.** Cuando se obtienen contribuciones tanto de personas como de organizaciones. Aunque antes de recibirlo se debe saber lo que se va a hacer con él.

5.20.2 Contratos indefinidos

Cuando se dirige un proyecto de software, sea de fuente abierta o propietario, se debe intentar mantener el tiempo máximo a los desarrolladores, ya que si se cambian cada poco tiempo le vendrá mal al proyecto, porque el programador que se vaya se llevara una credibilidad importante y muchas relaciones por parte de los usuarios, y se tendría que invertir más tiempo capacitándolo. También se debe tener en cuenta que un desarrollador nuevo no puede tener los mismos accesos al código que un desarrollador con experiencia, y tendrá que enviar parches hasta conseguir esos permisos. Además un programador nuevo puede empezar a preguntar sobre temas que ya se habían discutido antes.

Cuando un nuevo desarrollador se incorpore al proyecto, se recomienda capacitarlo mediante una participación supervisada, integrarlo con la comunidad de desarrollo público, comenzando con el seguimiento de errores y las tareas de limpieza, para que de esa manera pueda aprender el código base y ganar reputación en la comunidad. También se aconseja que al menos un desarrollador con experiencia pueda atenderlo cuando tenga algunas dudas respecto al software, y recomendarle que lea las discusiones de las listas de desarrollo.

5.20.3 Contrataciones

Los contratos se deben manejar con cuidado en los proyectos de fuente abierta, y se debe tratar que sean aceptados por la comunidad, aunque si el trabajo del contratista es bueno y cumple con las normas del proyecto no tendrá problemas [110]. Sin embargo, los contratos pueden presentar dos problemas:

- Si al contratista se le paga por hora, el proyecto podrá terminar pagando más de lo que se esperaba.
- Si se le paga una suma global, puede terminar haciendo más trabajo que el que puede permitirse.

Existen dos estrategias para evitar los dos problemas anteriores:

- Calcular la duración del proceso de discusión basado en la experiencia, las probabilidades de error y la base de ese contrato.
- Contratar exclusivamente para el lanzamiento de un parche.

Con cualquiera de las dos estrategias, el contrato en ningún momento puede requerir que el parche sea aceptado en el código, ya que eso significaría que se debe vender algo que no está en venta. En ese caso se debe tener un acto de confianza y recurrir a un acuerdo con la comunidad del proyecto.

Se recomienda contratar a uno de los desarrolladores del proyecto, preferiblemente un desarrollador encargado de hacer commit, ya que tiene conocimiento sobre el software, relación con los demás desarrolladores y se puede asegurar que su código es de buena calidad.

Los contratos se deben manejar públicamente para evitar problemas más adelante con los demás participantes del proyecto, en cuanto a una actitud diferente por parte del contratista tratando de discutir algo que era poco importante, y por las cuestiones de los demás participantes sobre su contrato.

5.20.4 Documentación y usabilidad

La documentación y la usabilidad son los dos puntos más débiles de un proyecto de fuente abierta [111]. Se recomienda contratar personas que se encarguen de estas tareas, y deben cumplir con una serie de características: que sean desarrolladores pero no expertos y que no estén en el desarrollo del mismo proyecto, es decir que tengan un buen conocimiento técnico a la hora de comunicarse con los demás desarrolladores pero que no estén especializados en el propio software, ya que tendrían problemas para describir la usabilidad desde un punto de vista externo y podría automatizar muchos pasos que para los usuarios no son tan obvios.

5.21 Comunicaciones

La capacidad para escribir puede ser la habilidad más importante en un ambiente de código abierto, incluso más importante que el talento para programar. Un buen programador con dificultades para comunicarse puede tener problemas para captar la atención de los demás, mientras que un mal programador con

habilidades para comunicarse puede coordinar y convencer a los demás. Es muy importante enfatizar con la audiencia, poder ver los correos y comentarios con el objetivo que ellos lo escriben y viceversa. Este apartado trata sobre la conducción de la comunicación y la forma para convertirla en una prioridad para todos [112].

5.21.1 Estructura y formato

Se recomienda usar una gramática estándar y una estructura narrativa coherente, escribir frases completas poniendo la primera letra de cada frase en mayúscula y usar separaciones de párrafo donde sea necesario [113]. Esto hará que el texto sea más legible, de manera que las personas puedan entender la idea expuesta y se lleven una buena impresión de la persona que la escribe. Existen algunas convenciones a la hora de redactar correos electrónicos:

- **Enviar correo solo de texto plano.** Se recomienda no exceder las 80 columnas de largo en el formato de las líneas, de manera que no se fuerce un estrechamiento de código cuando sea citado en otras respuestas.
- **Utilizar saltos de línea reales.** Se recomienda activar la opción para ver los saltos de línea reales a la hora de redactar un correo, ya que en algunas ocasiones está inhabilitado y podría dar un mal formato al texto.
- **Cuando se cite un correo de alguien, insertar la respuesta donde sea más apropiado.** Se recomienda escribir la respuesta encima o debajo del texto citado y eliminar las partes irrelevantes.
- **Construir el texto del nuevo correo con cuidado.** Se recomienda crear un nuevo correo, y no responder uno ya existente y cambiarle el asunto, de manera que no pueda filtrarse por el tema en común anterior.

5.21.2 Contenido

Existen algunos principios que pueden garantizar el buen contenido y a la vez pueden mantener a los lectores [114]:

- **Hacer las cosas fáciles para los lectores.** Habiendo tanta información alrededor del proyecto, se recomienda suministrarla a los usuarios de manera conveniente, como por ejemplo brindar la URL de un tema en particular para evitar que ellos lo hagan, resumir un tema complejo para su mejor entendimiento. Las personas, al ver trabajar de esta manera, terminan

imitando este estilo, y esto generará un incremento en la eficiencia del proyecto.

- **No acostumbrarse a la hipérbole.** Se recomienda evitar incrementar la importancia o gravedad del reporte de un fallo, ya que se puede perder credibilidad. Mientras que si se actúa con moderación, las personas lo tomarán con bastante seriedad.
- **Corregir dos veces.** Se recomienda revisar el texto del correo justo antes de enviarlo, ya que se puede perder el sentido narrativo. Se debe tener en cuenta que entre mejor estructurado se encuentre el texto, más usuarios lo leerán.

5.21.3 Tono

La mayoría de los foros técnicos suelen manejar sus discusiones de forma concisa, y no hay nada de malo en ello. No llevar un mensaje de bienvenida ni de despedida excepto su nombre es muy normal encontrarlo, por el contrario no se debe tomar a mal, ya que se está tomando la tarea de revisar la discusión y dar su opinión [115].

Claro está, que todo depende del contexto y de la persona. Por ejemplo si un usuario envía un correo excusándose por haber cometido un error, se recomienda responderlo de forma concisa y a la vez tratando de subir su ánimo. Si se está enviando un correo a un usuario exponiendo varios consejos para solucionar un error, se recomienda despedirse con un mensaje de buena suerte. Generalmente, se pueden utilizar emoticones en los mensajes para tranquilizar a los usuarios.

Se debe tener en cuenta que el ánimo y los sentimientos de los usuarios son muy importantes ya que afectan su productividad. Esto no quiere decir que se debe actuar como terapeuta personal de los usuarios, pero si se debe prestar atención al tono de los mensajes, que pueden convertirse en un beneficio, y en peor caso maleficio, para el proyecto a largo plazo.

5.21.4 No enviar correos sin propósito

Siempre hay que tener presente que habrán mas temas de los que un administrador pueda manejar y que todos los comentarios realizados allí no requieren ser respondidos [116]. Por lo general, en estos temas se manejan tres tipos de mensajes:

1. Mensajes de proposición.
2. Mensajes de apoyo u oposición frente a uno de los comentarios.
3. Mensajes de recapitulación.

Ninguno de estos mensajes requiere una respuesta de alguno de los administradores, ya que algún usuario puede llegar a decir lo mismo. Se recomienda a los administradores añadir comentarios solo cuando crean que se está proponiendo algo y nadie más se ha percatado de ello, cuando se necesitan aclarar ciertos criterios entre diferentes usuarios, y para agradecer a un usuario por algún comentario.

5.21.5 Temas productivos y temas improductivos

Cuando se tienen listas de correos muy concurridas, es necesario identificar los temas productivos y los improductivos [117]. Algunos de los síntomas de un tema improductivo son:

- Se repiten argumentos.
- Se incrementa el nivel de exageración y participación pero disminuye el interés.
- La mayoría de comentarios pertenecen a personas poco activas en el proyecto, mientras que los usuarios activos mantienen silencio.
- Muchas ideas se discuten sin un propósito claro.

Al identificar un tema improductivo, se debe tratar de cambiarlo para volverlo productivo. Se recomienda sugerir condiciones para promover progresos, guiar a la gente, claro está sin ser grosero y manejando un buen tono. Se debe tener un poco de paciencia, ya que al principio pueden parecer muchos correos pero pueden mencionar algo importante para el proyecto.

5.21.6 Gente difícil

La gente difícil puede definirse como grosera y maleducada. Pero más allá de esto, en un proyecto de software, la gente difícil es aquella que manipula y abusa en los procesos del proyecto, conllevando a pérdida de tiempo y energía de los demás usuarios. Un ejemplo es cuando la discusión que se está llevando a cabo no está lista para solucionarse y la persona calificada como difícil sigue ofreciendo

mas soluciones [118]. También cuando trata de retrasar un comentario en forma sumario, el cual contiene puntos o características que no le gustan.

Cuando se ha identificado a esta persona, se recomiendan una serie de pasos:

- Reunir evidencias. Pueden ser referencias a archivos públicos.
- Entablar una conversación privada con los demás administradores o directores y preguntarles lo que ellos observan, sin decirles aun lo que has identificado.
- Si los demás han identificado lo mismo, alguien se debe dirigir a la persona difícil. Manejando ante todo un buen tono y sin ser grosero, dirigirse a la persona mencionándole lo ocurrido y mostrándole las evidencias. Luego de esto, la persona podrá reformar su actitud, o en el peor caso seguir igual y se debe expulsar del proyecto.

5.22 Coordinando a los voluntarios

En este apartado se estudiará una serie técnicas que permitirán coordinar los diferentes voluntarios con los que se pueden contar durante el desarrollo del proyecto de fuente abierta [119].

5.22.1 Conseguir el máximo de voluntarios

Cuando un proyecto de fuente abierta es hecho público, lo más necesitado y deseado es conseguir voluntarios que colaboren al desarrollo del mismo, para esto es necesario conocer las motivaciones de las personas [120]. Entre estas motivaciones se encuentran el deseo de producir buen código, el desafío y valor educativo de trabajar en problemas difíciles, el deseo de trabajar con otras personas, y el deseo de dar y recibir respeto por parte de todo el grupo de trabajo colectivo.

5.22.2 Delegar

Delegar significa pedirle a otra persona que haga un trabajo, ya sea por eficiencia, importancia o por querer involucrar más a alguien en el proyecto [121]. Se debe tener en cuenta que al delegar a otra persona un trabajo, lo puede tomar como confianza depositada en él, un mayor grado de compromiso y responsabilidad. Cabe mencionar que la persona se puede sentir presionada, y por esta razón se debe manejar un buen tono, de manera que le sea fácil rechazar la oferta. Es importante saber que el delegar trabajo puede ser utilizado al revés, de manera

que una persona le delegue trabajo a un administrador o que el administrador se ofrezca para realizarlo.

5.22.3 Distinguir entre pedir y asignar

En algunas ocasiones, es normal suponer que una persona aceptara la delegación de un trabajo, por ejemplo cuando está a cargo de alguna tarea y ha tenido errores, se supone que lo tomara voluntariamente [122]. Sin embargo, hay ocasiones donde no se puede esperar a que alguien se haga cargo. Si se decide asignar la tarea a una persona porque es experta en el tema, esta podría sentirse incomoda y presionada. Se recomienda distribuir la carga de trabajo equitativamente y abrir un dialogo, y manejar la conversación con buen tono para hacer la petición, de manera que no se ejerza presión y se archive la conversación.

5.22.4 Supervisar después de delegar

Cuando se delega un trabajo a otra persona, se debe hacer seguimiento de la petición. La persona puede o no aceptar el trabajo; en el caso que no acepte el trabajo, entonces se cierra el tema y se busca otra alternativa para realizarlo; en el caso que si se acepte el trabajo, se debe supervisar la evolución del trabajo y hacer comentarios sobre el tema [123]. Si después de un tiempo la persona encargada no respondió nada sobre esta petición, se recomienda preguntar en los foros y las listas de correo si hay alguien interesado en el trabajo previamente asignado, y asegurar que no se recibió respuesta de la petición. Esto dará a entender a los usuarios que el administrador está atento a todo lo que pide.

5.22.5 Tratar cada usuario como posible voluntario

Siempre que se tope con un usuario, sin importar si es en la lista de correos o en los foros, se debe tener en cuenta que puede ser un posible voluntario [124]. Si el usuario presenta un informe de error o recomienda incluir una pregunta en el apartado de preguntas frecuentes, se le debe dar las gracias y ofrecerle la opción de solucionarlo o de incluir la pregunta en el apartado. Actuando de esta manera, permite a los demás usuarios tener presente que se pueden vincular al proyecto muy fácilmente.

5.22.6 Gerente de parches

En un proyecto de fuente abierta que recibe muchos parches, es muy difícil manejarlos y seguirlos todos. Algunas veces los usuarios que pertenecen a la lista de correo o al foro donde se discutieron las funcionalidades del parche, lo revisan y si encuentran un error lo reportan de inmediato, y el desarrollador vuelve y lo toma [125]. Este es un proceso que se repite hasta que no reportan más inconsistencias. En algunas ocasiones, cuando el parche se da por terminado, el desarrollador hace commit al proyecto, pero en otras no lo hace, debido a que no tiene tiempo o no quiere comprometerse.

El gerente de parches se encarga de asegurar que todos los parches registrados no pasen por alto. Debe vigilar cada discusión, foro y lista de correo, y estar pendiente de los parches en proceso de desarrollo. Cuando pase por un tema y encuentre que el parche está terminado y no se ha hecho commit, debe registrar que ya está terminado y brindar la referencia de la versión final a todos los temas que de alguno u otra manera están vinculados con el mismo. Cuando el parche no ha generado ninguna reacción, el gerente de parches debe preguntar si alguien lo revisara, y en caso que siga sin respuesta, se debe comentar como bueno o malo, con el fin de buscar una reacción por parte del desarrollador.

5.22.7 Gerente de traducción

La traducción puede referirse a dos cosas diferentes: traducción de la documentación del software a otros idiomas, o traducir el software en sí, como los errores en pantalla o mensajes de ayuda. Sabiendo que estas dos cosas son complejas, se debe tener un gerente de traducción [126].

El gerente de traducción se debe encargar de coordinar el grupo de traductores de diferentes idiomas, también debe asegurar que las traducciones no interfieran con el desarrollo, debe ser el representante de todos los traductores, y por ultimo debe encargarse de crear una lista de correo para cada equipo, de manera que los traductores puedan discutir su trabajo libremente.

5.22.8 Gerente de documentación

La documentación es una de las cosas más importantes del proyecto, y se debe tener siempre al día, es decir que es una tarea que perdurara siempre. Cada vez que se haga un cambio en el código, debe actualizarse la documentación para llevar una consistencia de lo que se está haciendo [127]. En esta caso, se debe

contar con un gerente de documentación, el se encargue de mantener la información actualizada o maneje un pequeño grupo dedicado a la tarea.

5.22.9 Gerente de errores

A medida que va creciendo la comunidad del proyecto y sus usuarios, al mismo tiempo va creciendo la cantidad informes de errores o problemas reportados. Por esta razón se recomienda contar con un gerente de errores, el cual se encarga de hacer una observación periódica en la base de datos de fallos buscando solucionar errores entrantes, en caso que no se haya diligenciado algún campo del formulario o porque el informe presentado es un duplicado de uno que ya existe. Por este motivo, el gerente de errores debe ser una persona que muy familiarizada con la base de datos de fallos desde un inicio [128]. También se pueden encargar de dirigir un informe de error al tema donde ciertamente pertenece de acuerdo a la lista de correo o foro del proyecto.

5.22.10 Gerente de preguntas frecuentes

El mantenimiento del documento de las preguntas frecuentes es muy difícil. Sin importar el tamaño que tenga, no se sabe cuál será la siguiente pregunta a incluir; y por ser un documento que se actualiza cada poco tiempo, puede llegar a ser desordenado e incoherente, incluso puede tener entradas duplicadas [129]. El gerente de preguntas frecuentes se encarga agrupar todas las preguntas por un mismo tema, también debe estar pendiente de las nuevas entradas en las listas de correo para agregar la pregunta en su respectivo tema y debe seguir toda la discusión con el fin de poder encontrar su solución.

5.22.11 Committers

Los committers son las personas que se encargan de hacer commit en el proyecto. Estas personas merecen una atención especial ya que ellos desempeñan una función absolutamente necesaria, y se puede decir que un proyecto les debe el éxito que alcance [130].

- **Elegir committers.** Se recomienda elegir committers una cualidad primordial, debe demostrar un buen juicio, es decir que sabe que debe asumir y que no [131]. Si una persona publica pequeños parches y arregla problemas simples de código, pero se aplican limpiamente, no contienen errores, esta consistente con el requerimiento y muestra un patrón claro,

entonces es un candidato perfecto para ser committer y se le puede hacer la oferta.

- **Revocando acceso de committers.** Se debe tratar de no buscar esta situación, ya que significaría perder tiempo de trabajo productivo [132]. Pero en el caso que se tenga que hacer, se debe abrir una discusión entre las personas encargadas de otorgar el derecho de hacer commit sin incluir a la persona implicada, ya que las personas no podrían hablar libremente y en el caso que se decida no revocarlo, entonces sentiría como enemigos a los que estuvieron a favor del movimiento. En algunas ocasiones, se decide dar una advertencia al committer, pero antes se debe deliberar igualmente.

6 APLICACIÓN DE LA GUIA DE RECOMENDACIONES

En este apartado se ofrecerán diversos ejemplos como soluciones a las recomendaciones planteadas en el capítulo anterior. Se debe tener en cuenta que algunas de las recomendaciones son aplicadas a lo largo del proyecto y dependen de la personalidad, organización y forma de expresión de cada persona; también dependen de la documentación de cada proyecto. Por lo tanto no se puede incluir un ejemplo para cada recomendación expuesta.

Por otro lado, se dará a conocer la solución escogida de cada recomendación de la guía para el proyecto de fuente abierta Zathuracode.

1. Escoger el nombre del proyecto.

El nombre del proyecto es Zathuracode, y aunque ya se había seleccionado con anterioridad, se puede concluir que cumple con la recomendación de la guía, la cual dice que se debe escoger un nombre fácil de recordar y fácil de pronunciar.

2. Declarar los objetivos.

La declaración de objetivos es parte de cada proyecto, y por ende no se pueden dar ejemplos. Por tal motivo solo se expondrán los objetivos del proyecto Zathuracode, de una manera que cumpla con la recomendación de la guía. De acuerdo a esto, la declaración de objetivos será:

“Construir un Generador de código para la plataforma JavaEE con ayuda de la comunidad, para permitir a los desarrolladores elegir entre diferentes arquitecturas basadas en estándares abiertos y patrones de diseño JavaEE.”

3. Declarar que el proyecto es libre.

Siguiendo la recomendación de la guía, la declaración indicando que el proyecto es libre, la cual se describe de manera directa y exponiendo su licencia, se muestra a continuación:

“Proyecto de fuente abierta licenciado bajo la licencia Apache 2.0.”

4. Lista de características y requerimientos.

Teniendo en cuenta la descripción del apartado, la lista de características y requerimientos se muestra de tal manera que se expongan sus compatibilidades y funcionalidades de acuerdo al sistema. Algunas de ellas del proyecto Zathuracode, se muestra en la siguiente lista:

- Compatible con Eclipse 3.6
- Compatible con MyEclipse 8.5
- Compatible con Primefaces:
 - Primefaces HibernateCore
 - Primefaces JPA
 - Primefaces Spring HibernateCore
 - Primefaces Spring JPA
- Compatible con Icefaces:
 - Icefaces HibernateCore
 - Icefaces JPA
 - Icefaces Spring HibernateCore
 - Icefaces Spring JPA

5. Estado del desarrollo.

En este apartado se listarán las diferentes versiones del proyecto Zathuracode con sus respectivas características. En cuanto a los objetivos a corto plazo, se puede indicar la necesidad de construir manuales para usuarios y desarrolladores, de manera que sea mucho más fácil el entendimiento para ellos. Las versiones que se listarán son:

- Zathuracode 3.0.0
- Zathura 2.1.1
- Zathura 2.1.0
- Zathura 2.0

6. Descargas.

Teniendo en cuenta la recomendación de la guía, cuando se empiezan a ofrecer versiones, se debe pensar en la forma de indicar el orden de lanzamientos, con el fin de diferenciar una versión vigente de una antigua. Basado en esto, se pueden brindar algunos ejemplos para escribirlos de una manera conveniente:

- MiProyecto.v1.0.0.0
- MiProyecto.v1.0.0.0.Estable
- MiProyecto.v1.0.0.0.Beta
- MiProyecto.v1.0.0.0.Alfa

- v1.0.0.0-MiProyecto
- v1.0.0.0.Estable-MiProyecto
- v1.0.0.0.Beta-MiProyecto
- v1.0.0.0.Alfa-MiProyecto

- v1.0.0-MiProyecto
- v1.0.0-MiProyecto-Estable
- v1.0.0-MiProyecto-Beta
- v1.0.0-MiProyecto-Alfa

- MiProyecto-v1.0.0-25.11.12
- MiProyecto-v1.0.0.Estable-23.11.12
- MiProyecto-v1.0.0.Beta-20.11.12
- MiProyecto-v1.0.0.Alfa-10.11.12

Estos ejemplos mencionados, indican de manera clara un orden de lanzamientos, empezando con el más reciente hasta el más antiguo. También incluyen un descriptor de la versión, la cual es señalada por el término 'Alfa', 'Beta' y 'Estable'. También se incluyó un ejemplo que contiene la fecha del lanzamiento, con el fin de aclarar aún más su orden.

Por otra parte, para el proyecto Zathuracode, actualmente se está usando un estándar que muestra el nombre del proyecto, el resultado del proyecto o el programa final, y el número de la versión. El estándar de versión para Zathuracode es:

- Zathuracode-eclipse-plugins-v3.0.0

7. Control de versiones y seguimiento de errores.

El sistema de control de versiones es muy importante para el proyecto, ya que permite manejar y equilibrar el código y el esfuerzo de los desarrolladores. Algunos de los sistemas de control de versiones que existen en el negocio y se puede acceder fácilmente a ellos son Mercurial, Git, SVN (Subversion), Bazaar, CVS (Concurrent Versions System), entre otros.

Para el proyecto Zathuracode, el control de versiones utilizado es SVN (Subversion).

Por otro lado, el sistema de seguimiento de errores también es muy importante, ya que permite llevar un control sobre los fallos del programa. Algunos de los sistemas de seguimiento de errores que existen en el mercado son BugZilla, Mantis, Debian Bug Tracking System (DBTS), RequestTracker (RT), entre otros. Para el proyecto Zathuracode, se decidió contar con el sistema Mantis.

8. Canales de comunicación.

Para este apartado, en el Proyecto Zathuracode se suministran correos electrónicos de los participantes, en los cuales se puedan enviar comentarios para hacer una retroalimentación del proyecto y para que los interesados puedan enviar sus sugerencias y solicitudes.

9. Pautas de desarrollo.

En este punto se ofrecen los enlaces a los diferentes sistemas y módulos de la página web del proyecto, en los cuales se pueden reportar fallos, sugerencias, solicitudes, realizar descargas de versiones y documentación.

En cuanto al proyecto Zathuracode se ofrecen los enlaces para el sistema de seguimiento de errores, y se suministran los correos electrónicos de los participantes del proyecto.

10. Documentación.

En el proyecto Zathuracode, en cuanto al apartado de documentación, que trata sobre ofrecer la documentación necesaria para los usuarios y los desarrolladores, se puede poner como un objetivo a corto plazo, ya que no se cuenta con esta

documentación. En el momento que se complete esta documentación, se deben ofrecer los enlaces para descargarla.

11. Ejemplos de salidas y capturas.

Es importante contar con al menos un ejemplo de funcionamiento del producto, ya que asegurara a los usuarios que funciona.

El proyecto Zathuracode cuenta con un video tutorial, el cual muestra la funcionalidad y las instrucciones para usar el producto.

12. Hosting enlatado.

Algunos de los sitios más importantes que ofrecen el hosting y su infraestructura gratuita son sourceforge.net, code.google.com, github.com, codeplex.com, entre otros.

Anteriormente, el proyecto Zathuracode se encontraba alojado en code.google.com.

Actualmente, basándose en la guía de recomendaciones, se decidió contar con un dominio propio y montar todo su sitio web. El dominio escogido fue www.zathuracode.org y la plataforma para gestionar su contenido fue wordpress.

13. Escogiendo una licencia.

En este apartado se mencionan las licencias “Haz lo que quieras” y la licencia “GPL”, pero aparte de estas existen otras en el mercado, como la MIT/X Window System License, la cual permite utilizar el código conjuntamente en software propietario; la BSD, la cual es similar a la anterior, exceptuando una clausula que obliga a mencionar el grupo propietario del código abierto; la licencia Apache que permite usar el software libremente, distribuirlo, modificarlo, y no requiere redistribuir el código fuente en próximas versiones.

Para el caso del proyecto Zathuracode, es licenciado bajo la licencia Apache License 2.0 y está expuesta en el sitio web del proyecto.

14. Manera de comunicarse.

Este apartado esta basado en la personalidad, organización y forma de expresión de cada persona o director de proyecto, y se aplica a largo de todo el proyecto, por tal motivo no se pueden dar algunos ejemplos.

En el caso de Zathuracode, se debe tener en cuenta que en este momento se esta adaptando su sitio web basado en esta guía, por lo tanto no se puede mencionar un caso en específico.

15. Anunciar.

Cuando se haya terminado de adaptar su sitio web y ofrezca toda la documentación necesaria para usuarios y desarrolladores, el paso a seguir será anunciarlo en freshmeat.net o freecode.com, el cual es uno de los lugares que más usuarios tiene, interesados en participar en algún proyecto. Un ejemplo para anunciar el proyecto Zathuracode puede ser:

“Contacto: dgomez@vortexbird.com

Nombre del proyecto: Zathuracode

Descripción:

Zathuracode es un generador de código de fuente abierta, que intenta tener todo el código generado lo mas limpio posible, para que los desarrolladores lo cambien según sus necesidades.

Su objetivo es permitir a los desarrolladores escoger entre diferentes arquitecturas basadas en estándares abiertos y patrones de diseño JavaEE.

Tecnologías:

- Primefaces: HibernateCore, JPA, Spring HibernateCore, Spring JPA.
- Icefaces: HibernateCore, JPA, Spring HibernateCore, Spring JPA.

Sitio web: <http://www.zathuracode.org>

D. Gómez”

16. Listas de correo.

A medida que va creciendo el proyecto y que vayan creciendo los participantes, ya sean usuarios o desarrolladores, es importante contar con un sistema que permita manejar las listas de correo.

En el caso del proyecto Zathuracode, el cual utiliza la plataforma para gestionar contenido WordPress, se pueden configurar algunos sistemas para manejar las listas de correo, como son: Mail list, Easy sign up, Benchmark email lite, entre otros.

17. Sistemas de chat en tiempo real.

Al igual que en el caso anterior, para contar con un sistema de chat es preferible que se tenga un gran numero de visitas al día para añadirlo al sitio web, ya que si no, parecerá un sitio web o un proyecto desierto. Algunos de los sistemas de chat que se pueden incluir en el proyecto Zathuracode pueden ser: User messages, Semi-private comments, PM for WordPress, WP-Live-Chat, entre otros.

Sin embargo, se incluyó un sistema de chat en tiempo real dentro del sitio web de Zathuracode, el cual es User Messages y permite la comunicación directa entre los participantes.

18. Wikis.

Los wikis permiten editar el contenido del sitio web con ayuda de los participantes permitidos para hacerlo. En el mercado existen algunos programas que incluyen un wiki: PHPWiki, TikiWiki, MoinMoin, entre otros.

En cuanto al proyecto Zathuracode, hasta el momento no se tiene planeado contar con un wiki para su sitio web, pero podría instalarse el plugin para WordPress llamado WordPress wiki.

19. Infraestructura social y política.

Este tema hace referencia a la organización personal del director del proyecto, por lo tanto no se brindaran ejemplos.

En cuanto al proyecto Zathuracode, esto se aplicará a medida que vaya pasando el tiempo y vaya creciendo el número de participantes.

20. Dinero.

De igual manera que el punto anterior, el dinero hace parte de la organización del director del proyecto y por esa razón no se pueden dar ejemplos.

Para el proyecto Zathuracode se manejarán los contratos, participaciones y donaciones a medida que se presenten.

21. Comunicaciones.

Este punto también depende de la personalidad, organización y forma de expresión de cada persona, y de la documentación de cada proyecto, por esa razón no se pueden ofrecer ejemplos.

En cuanto al proyecto Zathuracode, el tono se aplicará en cada conversación o discusión que se presente, y la documentación que se requiere completar se hará con el estándar solicitado.

22. Coordinando a los voluntarios.

Por ultimo, el tema de coordinar a los voluntarios, se va aplicando a medida que va creciendo el número de participantes activos del proyecto, con el fin de dividirlos por cargos y puedan tener clara su labor dentro del proyecto.

Hasta el momento, para el caso del proyecto de Zathuracode, los participantes están divididos por committers y propietarios.

7 LISTA DE CHEQUEO

Basado en las recomendaciones expuestas anteriormente, se propone una lista de chequeo, la cual consiste en crear un grupo de preguntas o tareas, que deben ser inspeccionadas en el proyecto, para verificar las características que se cumplen, y recordar los puntos que faltan por implementar.

Las tareas que se incluirán en la lista de chequeo son:

Escoger el nombre del proyecto, declarar los objetivos, declarar que el proyecto es libre, listar características, extraer requerimientos, informar el estado del desarrollo, establecer formato de descargas, utilizar un sistema control de versiones, utilizar un sistema de seguimiento de errores, utilizar listas de correo, ofrecer enlaces a foros, ofrecer documentación para usuarios y desarrolladores, ofrecer capturas de pantalla en ejemplos de funcionalidades, escoger sitio web o hosting, escoger una licencia, hacer uso y fomentar el buen tono, manejar acuerdos entre todos los usuarios, manejar mediante acuerdos el dinero, conseguir voluntarios, y establecer voluntarios por roles.

Por último, se hizo una plantilla de la lista de chequeo (ver ANEXO 1. Lista de chequeo).

8 CONCLUSIONES

- Para llevar un proyecto de fuente abierta al éxito no basta con solo ser un experto o tener un grupo de expertos en programación, por el contrario, se necesitan una serie de requisitos personales y tecnológicos que pueden ser difíciles de obtener debido a la disposición de la persona.
- Si se tiene un orden y una forma de trabajar establecida desde el principio, siguiendo cada recomendación de manera confiada, será más fácil de actualizar a medida que va pasando el tiempo.
- Al momento de iniciar un proyecto de fuente abierta, es mejor incluir la documentación y el empaquetado desde el mismo momento, sin importar el tiempo que se tarde en hacerlo, para evitar que crezca a medida que va evolucionando el proyecto.
- Es más difícil dirigir un proyecto de software fuente abierta que un proyecto de software propietario, ya que en el primero, las personas son libres de decidir el momento de trabajar en el proyecto y será más difícil de incentivarlos a hacerlo.
- Cuando se tenga planeado iniciar un proyecto de fuente abierta, primero se debe investigar por diferentes partes si existe un proyecto con las mismas características o parecidas, con el fin de evitar duplicar el trabajo que ya existe, o investigar las razones por las cuales no fue exitoso.
- Al escoger una licencia para aplicarla al proyecto de fuente abierta, se debe tener claro lo que se desea permitir con el software, y luego, mirar detalladamente cada licencia hasta encontrar la más conveniente.
- Desde el momento que se inicia un proyecto de fuente abierta es muy importante manejar y difundir un buen tono con todos los participantes, y mantener la armonía entre toda la comunidad, ya que esto atraerá a nuevos usuarios y desarrolladores, y mantendrá los que ya están vinculados.

REFERENCIAS BIBLIOGRÁFICAS

- [1]. <http://scalaria.co/open-source/>
- [2]. <http://www.laflecha.net/canales/softlibre/articulos/los-pecados-capitales-de-los-proyectos-open-source/>
- [3]. <http://producingoss.com/es/introduction.html>
- [4]. http://gluc.unicauca.edu.co/wiki/index.php/%C2%BFSoftware_Libre_o_Software_de_Fuente_Abierta%3F
- [5]. <http://es.scribd.com/doc/13588303/Movimiento-Del-Software-Libre>
- [6]. http://es.wikipedia.org/wiki/Software_libre
- [7]. <http://www.gnu.org/philosophy/free-sw.es.html>
- [8]. FSF (Fundación para el Software Libre): su objetivo es eliminar las restricciones sobre la copia, redistribución, entendimiento, y modificación de programas de computadoras.
http://es.wikipedia.org/wiki/Free_Software_Foundation
- [9]. GNU (GNU no es Unix): sistema operativo completamente libre.
<http://www.gnu.org/>
- [10]. Copyleft: permitir la libre distribución de copias y versiones modificadas de una obra u otro trabajo, exigiendo que los mismos derechos sean preservados en las versiones modificadas.
<http://www.gnu.org/copyleft/copyleft.es.html>
- [11]. <http://www.ilhn.com/datos/practicos/datosmartes/archives/003788.php>
- [12]. <http://es.wikipedia.org/wiki/Debian>
- [13]. http://es.wikipedia.org/wiki/Movimiento_del_software_de_codigo_abierto
- [14]. <http://www.opensource.org/docs/definition.php>,
<http://usemoslinux.blogspot.com/2010/04/diferencias-entre-el-software-libre-y.html>
- [15]. OSI (Iniciativa de fuente abierta): organización dedicada a la promoción del código abierto. <http://www.opensource.org/>
- [16]. <http://www.eoi.es/blogs/open/software-libre-vs-fuentes-abiertas/>
- [17]. <http://www.eclipse.org/org/>
- [18]. <http://www.linuxfoundation.org/about>
- [19]. <http://www.mozilla.org>
- [20]. <http://www.apache.org/>
- [21]. <http://www.opensource.org/>
- [22]. <http://www.python.org/psf/>
- [23]. <http://wikimediafoundation.org/wiki/Home>
- [24]. <http://www.gnome.org/foundation/>

- [25]. <http://www.xiph.org/>
- [26]. <http://www.kde.org/>
- [27]. <http://www.mozilla.org/es-ES/firefox/new/>
- [28]. http://es.wikipedia.org/wiki/Mozilla_Firefox
- [29]. <http://www.linux.org/>
- [30]. http://www.linux-es.org/sobre_linux
- [31]. <http://es.libreoffice.org/>
- [32]. <http://es.wikipedia.org/wiki/LibreOffice>
- [33]. <http://www.android.com/about/>
- [34]. <https://play.google.com/store>
- [35]. <http://es.wikipedia.org/wiki/Android>
- [36]. <http://www.mozilla.org/es-ES/thunderbird/>
- [37]. <http://es.wikipedia.org/wiki/Thunderbird>
- [38]. <http://www.videolan.org/vlc/>
- [39]. http://es.wikipedia.org/wiki/VLC_media_player
- [40]. <http://www.gimp.org/>
- [41]. <http://es.wikipedia.org/wiki/GIMP>
- [42]. <http://es.clamwin.com/>
- [43]. http://www.filehippo.com/es/download_clamwin/history/4/
- [44]. <http://www.kde.org/>
- [45]. <http://es.wikipedia.org/wiki/KDE>
- [46]. <http://www.postgresql.org.es/>
- [47]. <http://es.wikipedia.org/wiki/PostgreSQL>
- [48]. <http://producingoss.com/es/index.html>
- [49]. <http://producingoss.com/es/getting-started.html#choosing-a-name>
- [50]. <http://producingoss.com/es/getting-started.html#mission-statement>
- [51]. <http://producingoss.com/es/getting-started.html#state-freedom>
- [52]. <http://producingoss.com/es/getting-started.html#features-and-requirements>
- [53]. <http://producingoss.com/es/getting-started.html#development-status>
- [54]. <http://producingoss.com/es/getting-started.html#downloads>
- [55]. <http://producingoss.com/es/development-cycle.html#release-numbering>
- [56]. <http://producingoss.com/es/development-cycle.html#release-number-components>
- [57]. <http://producingoss.com/es/packaging.html>

- [58]. <http://producingoss.com/es/getting-started.html#vc-and-bug-tracker-access>
- [59]. <http://producingoss.com/es/vc.html>
- [60]. <http://producingoss.com/es/vc.html#vc-vocabulary>
- [61]. <http://producingoss.com/es/vc.html#vc-choosing>
- [62]. <http://producingoss.com/es/vc.html#version-everything>
- [63]. <http://producingoss.com/es/vc.html#vc-browsing>
- [64]. <http://producingoss.com/es/vc.html#commit-emails>
- [65]. <http://producingoss.com/es/vc.html#branches>
- [66]. <http://producingoss.com/es/vc.html#vc-singularity>
- [67]. <http://producingoss.com/es/vc.html#vc-authz>
- [68]. <http://producingoss.com/es/bug-tracker.html>
- [69]. <http://producingoss.com/es/bug-tracker.html#bug-tracker-mailing-list-interaction>
- [70]. <http://producingoss.com/es/bug-tracker.html#bug-filtering>
- [71]. <http://producingoss.com/es/getting-started.html#communications-channels>
- [72]. <http://producingoss.com/es/growth.html>
- [73]. <http://producingoss.com/es/getting-started.html#developer-guidelines>
- [74]. <http://producingoss.com/es/written-rules.html>
- [75]. <http://producingoss.com/es/getting-started.html#documentation>
- [76]. <http://producingoss.com/es/getting-started.html#documentation-availability>
- [77]. <http://producingoss.com/es/getting-started.html#developer-documentation>
- [78]. <http://producingoss.com/es/getting-started.html#example-output>
- [79]. <http://producingoss.com/es/getting-started.html#starting-with-canned-hosting>
- [80]. <http://producingoss.com/es/license-quickstart.html>
- [81]. <http://producingoss.com/es/license-quickstart.html#license-quickstart-non-gpl>
- [82]. <http://producingoss.com/es/license-quickstart.html#license-quickstart-gpl>
- [83]. <http://producingoss.com/es/license-quickstart.html#license-quickstart-applying>
- [84]. <http://producingoss.com/es/setting-tone.html>
- [85]. <http://producingoss.com/es/setting-tone.html#avoid-private-discussions>

- [86]. <http://producingoss.com/es/growth.html#using-archives>
- [87]. <http://producingoss.com/es/setting-tone.html#prevent-rudeness>
- [88]. <http://producingoss.com/es/setting-tone.html#code-review>
- [89]. <http://producingoss.com/es/announcing.html>
- [90]. Feed RSS: Really Simple Syndication. Se utiliza para difundir información actualizada frecuentemente a usuarios que se han suscrito a la fuente de contenidos. <http://es.wikipedia.org/wiki/RSS>
- [91]. <http://producingoss.com/es/publicity.html>
- [92]. <http://producingoss.com/es/publicity.html#security>
- [93]. <http://producingoss.com/es/publicity.html#security-receiving>
- [94]. <http://producingoss.com/es/publicity.html#security-finding-a-fix>
- [95]. <http://producingoss.com/es/publicity.html#security-prenotification>
- [96]. <http://producingoss.com/es/publicity.html#security-announcing>
- [97]. <http://producingoss.com/es/mailling-lists.html>
- [98]. <http://producingoss.com/es/mailling-lists.html#spam-prevention>
- [99]. <http://producingoss.com/es/mailling-lists.html#archiving>
- [100]. <http://producingoss.com/es/irc.html>
- [101]. <http://producingoss.com/es/irc.html#irc-archiving>
- [102]. <http://producingoss.com/es/wikis.html>
- [103]. <http://producingoss.com/es/social-infrastructure.html>
- [104]. <http://producingoss.com/es/social-infrastructure.html#benevolent-dictator>
- [105]. <http://producingoss.com/es/consensus-democracy.html>
- [106]. <http://producingoss.com/es/consensus-democracy.html#voting>
- [107]. <http://producingoss.com/es/money.html>
- [108]. <http://producingoss.com/es/money.html#types-of-involvement>
- [109]. <http://producingoss.com/es/long-term-developers.html>
- [110]. <http://producingoss.com/es/contracting.html>
- [111]. <http://producingoss.com/es/funding-non-programming.html#subsidize-documentation-usability>
- [112]. <http://producingoss.com/es/communications.html>
- [113]. <http://producingoss.com/es/communications.html#structure-and-formatting>

- [114]. <http://producingoss.com/es/communications.html#writing-content>
- [115]. <http://producingoss.com/es/communications.html#writing-tone>
- [116]. <http://producingoss.com/es/common-pitfalls.html#post-with-purpose>
- [117]. <http://producingoss.com/es/common-pitfalls.html#productive-threads>
- [118]. <http://producingoss.com/es/difficult-people.html>
- [119]. <http://producingoss.com/es/managing-volunteers.html>
- [120]. <http://producingoss.com/es/managing-volunteers.html#volunteers>
- [121]. <http://producingoss.com/es/managing-volunteers.html#delegation>
- [122]. <http://producingoss.com/es/managing-volunteers.html#delegation-assignment>
- [123]. <http://producingoss.com/es/managing-volunteers.html#delegation-followup>
- [124]. <http://producingoss.com/es/managing-volunteers.html#users-to-volunteers>
- [125]. <http://producingoss.com/es/share-management.html#patch-manager>
- [126]. <http://producingoss.com/es/share-management.html#translation-manager>
- [127]. <http://producingoss.com/es/share-management.html#documentation-manager>
- [128]. <http://producingoss.com/es/share-management.html#issue-manager>
- [129]. <http://producingoss.com/es/share-management.html#faq-manager>
- [130]. <http://producingoss.com/es/committers.html>
- [131]. <http://producingoss.com/es/committers.html#choosing-committers>
- [132]. <http://producingoss.com/es/committers.html#revoking-committers>

ANEXOS

ANEXO 1. Lista de chequeo

LISTA DE CHEQUEO PROYECTO DE FUENTE ABIERTA

Nombre proyecto: _____

	TAREAS	Ejecutado
1	Escoger el nombre del proyecto	
2	Declarar los objetivos	
3	Declarar que el proyecto es libre	
4	Listar características	
5	Extraer requerimientos	
6	Informar el estado del desarrollo	
7	Establecer formato de descargas	
8	Utilizar un sistema de control de versiones	
9	Utilizar un sistema de seguimiento de errores	
10	Utilizar listas de correo, salas de chat, foros	
11	Ofrecer enlaces a foros, instrucciones y ayudas	
12	Ofrecer documentación para usuarios y desarrolladores	
13	Ofrecer capturas de pantalla en ejemplos de funcionalidades	
14	Escoger sitio web o hosting	
15	Escoger una licencia	
16	Hacer uso y fomentar el buen tono	
17	Manejar un historial de conversaciones	
18	Anunciar y publicar el proyecto	
19	Manejar acuerdos entre todos los usuarios	
20	Manejar mediante acuerdos el dinero	
21	Conseguir voluntarios	

22	Establecer voluntarios por roles	
----	----------------------------------	--

Diligenciado por: _____