PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR
FACULTAD DE INGENIERÍA
ESCUELA DE SISTEMAS

DISERTACIÓN DE GRADO PREVIA A LA OBTENCION DEL TITULO DE INGENIERÍA EN SISTEMAS Y CIENCIAS DE LA COMPUTACIÓN

IMPLEMENTACIÓN DE LA GESTIÓN DEL SERVICIO EN BASE AL MARCO DE TRABAJO DE ITIL V3 DIRIGIDO A LA GESTIÓN DE BUGS Y ERRORES EN PROYECTOS DE IMPLEMENTACIÓN DE SOFTWARE EN EMPRESAS DESARROLLADORAS, CASO DE ESTUDIO EMPRESA "YAGE".

ESTEBAN ANDRÉS VILLACÍS PONCE
FABIÁN SANTIAGO GARCÍA BAUTISTA

QUITO, 2014

ÍNDICE

1. TEMA

IMPLEMENTACIÓN DE LA GESTIÓN DEL SERVICIO EN BASE AL MARCO DE TRABAJO DE ITIL® V3 DIRIGIDO A LA GESTIÓN DE BUGS Y ERRORES EN PROYECTOS DE IMPLEMENTACIÓN DE SOFTWARE EN EMPRESAS DESARROLLADORAS, CASO DE ESTUDIO EMPRESA "YAGE".

2. INTRODUCCIÓN

ITIL® es una metodología muy utilizada por la mayoría de empresas públicas y privadas a nivel mundial, donde exista la gestión de las tecnologías de la información y la comunicación (TICs), es por esto nuestro interés en desarrollar una herramienta que permita a las empresas desarrolladoras de software, que pertenecen a las PYMES, gestionar sus servicios, proveedores, clientes y gestión de mejora continua de una manera sencilla y orientada a la mejora de la calidad en el software que desarrollan.

El propósito de esta herramienta es brindar los beneficios de ITIL® V3 sin que estos se conviertan en una traba burocrática para las empresas desarrolladoras de software y a la vez demostrar que sí se puede integrar las buenas prácticas de ITIL® a los procesos de las pequeñas y medianas empresas, caso de estudio empresa YAGE.

Al momento de orientar este software para convertirse en una herramienta de gestión de bugs[1] y errores en los proyectos que manejan estas medianas y pequeñas empresas incluyendo ciertos beneficios que consideramos indispensables para la mejora continua de la calidad del desarrollo de software logramos desarrollar un software intuitivo que creemos ayudará a mejorar la calidad en los proyectos de desarrollo de software.

Para este desarrollo hemos conseguido el apoyo de una renombrada empresa local de desarrollo de software y soluciones digitales, "YAGE" a la cual citaremos en el desarrollo de esta disertación. Creemos que al tener un caso de estudio con una

[1] Es un error o fallo en un programa de computador o sistema de software que desencadena un resultado indeseado. Los programas que ayudan a la detección y eliminación de errores de programación de software son denominados depuradores (debuggers), como la realizada en esta disertación.

empresa de soluciones digitales podemos ayudar con nuestra idea a varios otros negocios locales que necesitan de una guía para organizar sus procesos y mejorar la calidad del software que desarrollan sin que esta herramienta agregue complejidad a los procesos de desarrollo de software en este tipo de empresas.

CAPÍTULO I: LA EMPRESA

1.1 Historia

La empresa YAGE, se ha consolidado en el mercado nacional líder del país en el desarrollo de Soluciones Globales en Internet y Medios Interactivos. Combinando el mejor talento humano y las oportunidades generadas por la era digital, desarrollamos proyectos competitivos a nivel mundial para las principales empresas nacionales y multinacionales del país. Las principales fortalezas que nos han permitido posicionarnos como líderes en el país y proyectarnos a mercados internacionales han sido: la experiencia en diversos sectores, el capital humano especializado y los altos estándares de calidad aplicados en cada uno de los proyectos. Enfocamos nuestros proyectos desde una perspectiva estratégica y no técnica lo cual permite generar resultados tangibles en nuestros clientes.

1.2 Misión[2]

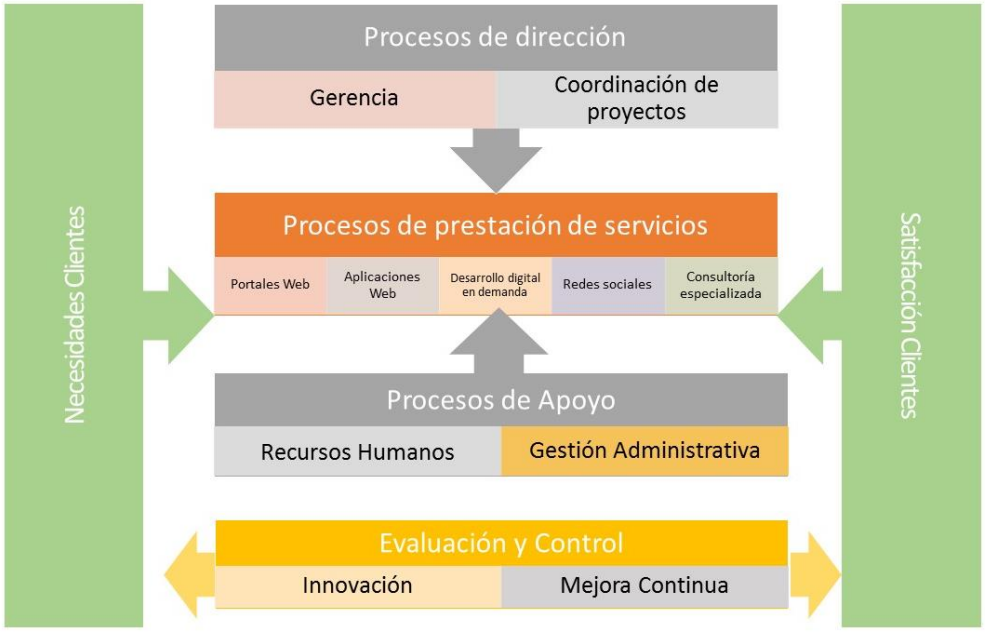
Brindar un enfoque diferente al Marketing Digital. Plantear soluciones Integrales a sus socios y clientes, desde la estrategia digital para la generación de resultados efectivos en el negocio, la planificación de medios, hasta la creación de plataformas y campañas innovadoras basadas en una creatividad al servicio de los objetivos del cliente y orientada a la mejor experiencia con el usuario.

1.3 Mapa de Procesos

Los procesos que la empresa realiza día a día con el objetivo de cumplir con su misión y visión son los siguientes.

[2] Información tomada del sitio web de la empresa YAGE.

Ilustración 1 - Mapa de procesos

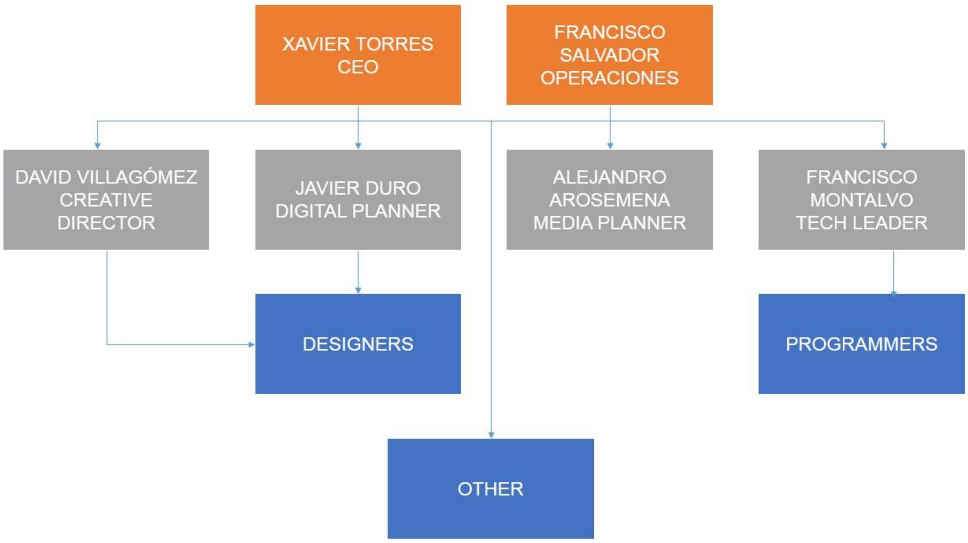


Fuente: YAGE
Elaborado por: Esteban Villacis y Santiago García

1.4 Organigrama

Se ha estructurado el orgánico funcional de la empresa YAGE, con las funciones y responsabilidades de los profesionales, facilitado por la empresa.

Ilustración 2 – Organigrama de la empresa YAGE



Fuente: YAGE[3]
Elaborado por: Esteban Villacis y Santiago García

1.5 Giro de Negocio

YAGE se dedica a brindar los siguientes servicios:

- Portales Empresariales
- Aplicaciones Web
- Estrategia Digital
    - Campañas de posicionamiento de marca en redes sociales
    - Experiencia digital de marcas
    - Consultoría y estadísticas de cuentas en redes sociales
- Consultoría Especializada

[3] Organigrama: Tomado de la página web informativa en construcción de la empresa YAGE – http://yage.yage.ec

CAPÍTULO II: MARCO TEÓRICO

2.1 Generalidades de ITIL® V3

2.1.1 Introducción

ITIL® es un conjunto de publicaciones de las mejores prácticas de las TICs donde se definen la estructura y habilidades requeridas por una organización para la gestión de las TIC.

En cada una de las publicaciones se especifican procedimientos y prácticas que ayudarán a mejorar la calidad de los servicios de TI y ofrecen un marco de respuesta eficiente ante problemas generados con una metodología de mejora continua que es el marco de trabajo de ITIL®. (Cartlidge, et al., 2012)

2.1.2 Descripción del Proyecto

El presente Proyecto tiene como objetivo la automatización de las fases de estrategia del servicio y diseño del servicio, basado en el marco de trabajo de ITIL® V3.

Entre los procesos que el proyecto contemplará están los de la administración del portafolio de servicios, administración del nivel del servicio, administración de la disponibilidad, administración de la continuidad del servicio y administración de proveedores todos estos procesos serán orientados a la gestión de bugs y/o errores dentro de los proyectos o sistemas de la empresa caso de estudio de la presente disertación.

2.2 ITIL® – Gestión del Servicio

2.2.1 ITIL® V3

El marco de trabajo de lTIL® está basado en el Ciclo de Vida del Servicio, el mismo que se encuentra conformado por 5 disciplinas principales: Estrategia del Servicio, Diseño del Servicio, Transición del Servicio, Operación del Servicio y Mejora

Continua del Servicio. La aplicación del Ciclo de Vida del servicio, permite que se provean servicios tecnológicos alineados a las necesidades del negocio, permitiendo mostrar valor para el negocio y retorno de inversión, además de que posibilita al departamento de Tl solventar necesidades específicas de la Operación tecnológica de la organización. (Arraj, 2013)

2.2.2 Introducción de la Gestión de Servicio

Para definir claramente la Gestión del Servicio basado en ITIL®, es importante conocer los términos inmersos en este tópico.

La Gestión es la administración de una o varias operaciones, donde se pretende obtener los mejores resultados dentro de una operación de cualquier índole. Por otro lado el servicio es el conjunto de actividades, beneficios o satisfacciones que proporcionan cierto valor a quien lo recibe, sin que el cliente asuma los riesgos y costos asociados. El marco de trabajo ITIL® V3, en esencia es la Gestión de la totalidad de los Servicios de TI, es decir administrar, supervisar, medir e innovar cada una de sus operaciones, para de este modo inyectar calidad en todas los servicios que se brinden a los clientes.

2.2.3 ITIL® en Empresas

Con la acelerada evolución tecnológica las empresas se han visto obligadas a automatizar la mayoría de sus procesos. La tendencia nos muestra que las organizaciones buscan una automatización total de sus procesos, hecho que sería imposible si se tiene un mal manejo de las TICs. Es por eso que en la actualidad la gerencia de tecnología no es la única preocupada por mantener un control sobre los servicios que brindan las TICs sino también todas las áreas dentro de una organización.

Como ejemplo podemos nombrar muchas empresas, ya sean empresas de TIC o de otra índole, que ya han implementado ITIL® para la mejora continua de su gestión de las TIC, pero existe siempre controversia cuando hablamos de pequeñas y medianas empresas. Muchas veces se dice que ITIL® solo tiene utilidad para

empresas grandes, donde hay extensos recursos disponibles para implementar y manejar tales procesos. Sin embargo, es un hecho que las ideas que guían ITIL® son valiosas para empresas de cualquier tamaño, desde las corporaciones grandes hasta negocios pequeños o medianos. Lo importante es aprovechar los beneficios de los principios de ITIL®, concentrándose en la viabilidad y relevancia durante su implementación. (Process Map, 2013)

2.2.4 Ciclo de Vida

ITIL® v3 estructura la gestión de los servicios TI sobre el concepto de Ciclo de Vida de los Servicios. Este enfoque tiene como objetivo ofrecer una visión global de la vida de un servicio desde su diseño hasta su eventual abandono o cierre sin por ello ignorar los detalles de todos los procesos y funciones involucrados en la eficiente prestación del mismo.

El Ciclo de Vida del Servicio consta de cinco fases que se corresponden con los nuevos libros de ITIL®:

1. Estrategia del Servicio: propone tratar la gestión de servicios no sólo como una capacidad sino como un activo estratégico.
2. Diseño del Servicio: cubre los principios y métodos necesarios para transformar los objetivos estratégicos en portafolios de servicios y activos.
3. Transición del Servicio: cubre el proceso de transición para la implementación de nuevos servicios o su mejora.
4. Operación del Servicio: cubre las mejores prácticas para la gestión del día a día en la operación del servicio.
5. Mejora Continua del Servicio: proporciona una guía para la creación y mantenimiento del valor ofrecido a los clientes a traces de un diseño, transición y operación del servicio optimizado. (Osiatis I. , El ciclo de vida de los servicios de TI, n.d.) (Arraj, 2013) (Cartlidge, et al., 2012) (The Stationery Office, 2011)

Ilustración 3 - ITIL®

Fuente: TCP – UST Global
Elaborado por: TCP – UST Global

2.2.5 Estrategia de Servicio

La fase de Estrategia del Servicio es central al concepto de Ciclo de vida del servicio y tiene como principal objetivo convertir la Gestión del Servicio en un activo estratégico.

Para conseguir este objetivo es imprescindible determinar en primera instancia qué servicios deben ser prestados y por qué han de ser prestados desde la perspectiva del cliente y el mercado. La fase de Estrategia del Servicio es el eje que permite que las fases de Diseño, Transición y Operación del servicio se ajusten a las políticas y visión estratégica del negocio.

Una correcta implementación de la estrategia del servicio va más allá del ámbito puramente TI y requiere un enfoque multidisciplinar que ayude a responder cuestiones tales como: Servicios a prestar, cómo crear valor en los servicios, resultados esperados, priorización de servicios, identificación de la competencia, etc. (Osiatis I. , n.d.) (Great Britain. Cabinet Office, 2011)

2.2.6 Diseño de Servicio

La principal misión de la fase de Diseño del Servicio es la de diseñar nuevos servicios o modificar los ya existentes para su incorporación al catálogo de servicios y su paso al entorno de producción.

El Diseño del Servicio debe seguir las directrices establecidas en la fase de Estrategia y debe a su vez colaborar con ella para que los servicios diseñados: Se adecuen a las necesidades del mercado, sean eficientes en costes y rentables, cumplan los estándares de calidad adoptados y aporten valor a clientes y usuarios. (Osiatis I. , Diseño para los servicios TI, n.d.) (Great Britain. Cabinet Office, 2011)

2.3 Buenas prácticas y herramientas para la implementación

2.3.1 Introducción al Extreme Programming (XP)

2.3.1.1 Definición de Extreme Programming (XP)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (Universidad Union Simon Bolivar, 2015)

2.3.1.2 Ciclo de Vida de XP

a) Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Estas historias de usuario a las que nos referiremos en el ciclo de vida de XP hacen referencia a la forma empírica o coloquial, si se quiere, que usan los usuarios o clientes para dar a conocer el proceso que realizan día a día y el que se quiere automatizar. Así mismo hay que tener en cuenta que estas historias son muy importantes dentro del ciclo de vida ya que nos permiten un entendimiento rápido y claro del proceso que se pretende automatizar.

Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología. (Wells, n.d.)

b) Planificación de la Entrega (Release)

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la “velocidad” de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar

basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación. (Wells, n.d.)

c) Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores. (Wells, n.d.)

d) Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas

que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento). (Wells, n.d.)

e) Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura. (Wells, n.d.)

f) Muerte del Proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo. (Wells, n.d.)

2.3.1.3 Justificación de XP

Elegimos programación extrema porque nos permite un ágil desarrollo de software, apostando por la prueba y error, enfocado a la extensa demanda de cambio de parecer del cliente concerniente a sus de requerimientos, y que aun así nos obliga a mantener un código limpio y de fácil legibilidad para personas externas. Adicionalmente quisiéramos mencionar que además de utilizar una metodología ágil de desarrollo encontramos muy útil un concepto muy utilizado que es combinar varias metodologías ágiles para el desarrollo de software de esta manera se puede sacar el mayor provecho de las fortalezas de cada una, como

por ejemplo podemos mencionar que en el desarrollo de esta disertación utilizamos “sprints”[4] cortos de desarrollo que son muy utilizados en SCRUM[5] pero al mismo tiempo mantuvimos el ciclo de vida de XP lo cual nos dio muy buenos resultados al momento de completar desarrollos en cortos tiempos. (Kniberg, 2007)

2.3.2 Software Libre

2.3.2.1 Licencias del Software Libre

Licencias GPL

Una de las más utilizadas es la Licencia Pública General de GNU (GNU GPL). El autor conserva los derechos de autor (copyright), y permite la redistribución y modificación bajo términos diseñados para asegurarse de que todas las versiones modificadas del software permanecen bajo los términos más restrictivos de la propia GNU GPL. Esto hace que sea imposible crear un producto con partes no licenciadas GPL: el conjunto tiene que ser GPL.

Es decir, la licencia GNU GPL posibilita la modificación y redistribución del software, pero únicamente bajo esa misma licencia. Y añade que si se reutiliza en un mismo programa código "A" licenciado bajo licencia GNU GPL y código "B" licenciado bajo otro tipo de licencia libre, el código final "C", independientemente de la cantidad y calidad de cada uno de los códigos "A" y "B", debe estar bajo la licencia GNU GPL.

En la práctica esto hace que las licencias de software libre se dividan en dos grandes grupos, aquellas que pueden ser mezcladas con código licenciado bajo GNU GPL (y que inevitablemente desaparecerán en el proceso, al ser el código resultante licenciado bajo GNU GPL) y las que no lo permiten al incluir mayores u otros requisitos que no contemplan ni admiten la GNU GPL y que por lo tanto no pueden ser enlazadas ni mezcladas con código gobernado por la licencia GNU GPL. (wikipedia, n.d.)

[4] Sprint, en la metodología de SCRUM es una manera de realizar ciclos de desarrollo cortos donde estos se puedan repetir completando el trabajo como en iteraciones de código.
[5] SCRUM, es una metodología ágil de desarrollo basada en varias iteraciones del ciclo de desarrollo de una aplicación.

Licencias AGPL

La Licencia Pública General de Affero (en inglés Affero General Public License, también Affero GPL o AGPL) es una licencia copyleft derivada de la Licencia Pública General de GNU diseñada específicamente para asegurar la cooperación con la comunidad en el caso de software que corra en servidores de red.

La Affero GPL es íntegramente una GNU GPL con una cláusula nueva que añade la obligación de distribuir el software si éste se ejecuta para ofrecer servicios a través de una red de ordenadores.

La Free Software Foundation recomienda que el uso de la GNU AGPLv3 sea considerado para cualquier software que usualmente corra sobre una red.8 (wikipedia, n.d.)

Licencias Estilo BSD

Llamadas así porque se utilizan en gran cantidad de software distribuido junto a los sistemas operativos BSD. El autor, bajo tales licencias, mantiene la protección de copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación, incluso si dichos trabajos tienen propietario. Son muy permisivas, tanto que son fácilmente absorbidas al ser mezcladas con la licencia GNU GPL con quienes son compatibles. Puede argumentarse que esta licencia asegura “verdadero” software libre, en el sentido que el usuario tiene libertad ilimitada con respecto al software, y que puede decidir incluso redistribuirlo como no libre. Otras opiniones están orientadas a destacar que este tipo de licencia no contribuye al desarrollo de más software libre (normalmente utilizando la siguiente analogía: "una licencia BSD es más libre que una GPL si y sólo si se opina también que un país que permita la esclavitud es más libre que otro que no la permite"). (wikipedia, n.d.)

Licencias Estilo MPL y Derivadas

Esta licencia es de Software Libre y tiene un gran valor porque fue el instrumento que empleó Netscape Communications Corp. para liberar su Netscape Communicator 4.0 y empezar ese proyecto tan importante para el mundo del Software Libre: Mozilla. Se utilizan en gran cantidad de productos de software libre de uso cotidiano en todo tipo de sistemas operativos. La MPL es Software Libre y promueve eficazmente la colaboración evitando el efecto "viral" de la GPL (si usas código licenciado GPL, tu desarrollo final tiene que estar licenciado GPL). Desde un punto de vista del desarrollador la GPL presenta un inconveniente en este punto, y lamentablemente mucha gente se cierra en banda ante el uso de dicho código. No obstante la MPL no es tan excesivamente permisiva como las licencias tipo BSD. Estas licencias son denominadas de copyleft débil. La NPL (luego la MPL) fue la primera licencia nueva después de muchos años, que se encargaba de algunos puntos que no fueron tomados en cuenta por las licencias BSD y GNU. En el espectro de las licencias de software libre se la puede considerar adyacente a la licencia estilo BSD, pero perfeccionada. (wikipedia, n.d.)

Copyleft

Hay que hacer constar que el titular de los derechos de autor (copyright) de un software bajo licencia copyleft puede también realizar una versión modificada bajo su copyright original, y venderla bajo cualquier licencia que desee, además de distribuir la versión original como software libre. Esta técnica ha sido usada como un modelo de negocio por una serie de empresas que realizan software libre (por ejemplo MySQL); esta práctica no restringe ninguno de los derechos otorgados a los usuarios de la versión copyleft.

En España, toda obra derivada está tan protegida como una original, siempre que la obra derivada parta de una autorización contractual con el autor. En el caso genérico de que el autor retire las licencias "copyleft", no afectaría de ningún modo a los productos derivados anteriores a esa retirada, ya que no tiene efecto retroactivo. En términos legales, el autor no tiene

derecho a retirar el permiso de una licencia en vigencia. Si así sucediera, el conflicto entre las partes se resolvería en un pleito convencional. (wikipedia, n.d.)

2.3.2.2 Introducción a PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga. (PHP.net, n.d.)

2.3.2.3 Justificación de PHP

Se usó PHP porque primero necesitábamos que nuestro software este enfocado hacia un Navegador Web, teniendo como ventaja el no instalar en cada computador en el que se vaya a utilizar. También lo elegimos porque es muy sencillo de aprender y de utilizar, así como también es muy potente para grandes tareas del lado del servidor.

2.3.3 Programación Orientada a Objetos

2.3.3.1 Características de P.O.O

Abstracción

Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un

"agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos, y, cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar. (wikipedia, n.d.)

Encapsulamiento

Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente. (wikipedia, n.d.)

Modularidad

Se denomina modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos. Al igual que la encapsulación, los lenguajes soportan la modularidad de diversas formas. (wikipedia, n.d.)

Principio de Ocultación

Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las

propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas; solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no puedan cambiar el estado interno de un objeto de manera inesperada, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos. (wikipedia, n.d.)

Polimorfismo

Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O, dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++. (wikipedia, n.d.)

Herencia

Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se

dice que hay herencia múltiple; siendo de alta complejidad técnica por lo cual suele recurrirse a la herencia virtual para evitar la duplicación de datos. (wikipedia, n.d.)

Recolección de Basura

La recolección de basura o garbage collection es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando. En la mayoría de los lenguajes híbridos que se extendieron para soportar el Paradigma de Programación Orientada a Objetos como C++ u Object Pascal, esta característica no existe y la memoria debe desasignarse expresamente. (wikipedia, n.d.)

2.3.4 Herramientas

2.3.4.1 Sobre Netbeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE[6] es un producto libre y gratuito sin restricciones de uso. (Wikipedia, n.d.)

2.3.4.2 Sobre MySQL

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

[6] IDE: Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas

Por un lado se ofrece bajo la GNU GPL[7] para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C[8]. (Wikipedia, n.d.)

2.3.4.3 Sobre Kohana

Kohana es un framework para aplicaciones web para PHP5 que implementa el patrón de Modelo Vista Controlador Jerárquico (HMVC). Sus principales objetivos se basan en ser seguro, ligero, y fácil de utilizar. (Wikipedia, n.d.)

[7] La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License (o simplemente sus siglas del inglés GNU GPL) es la licencia más ampliamente usada en el mundo del software y garantiza a los usuarios finales la libertad de usar, estudiar, compartir (copiar) y modificar el software

[8] ANSI C es un estándar publicado por el Instituto Nacional Estadounidense de Estándares (ANSI), para el lenguaje de programación C. Se recomienda a los desarrolladores de software en C que cumplan con los requisitos descritos en el documento para facilitar así la portabilidad del código.

CAPÍTULO III: DISEÑO DE LA APLICACIÓN

3.1 Introducción y Estado ITIL®

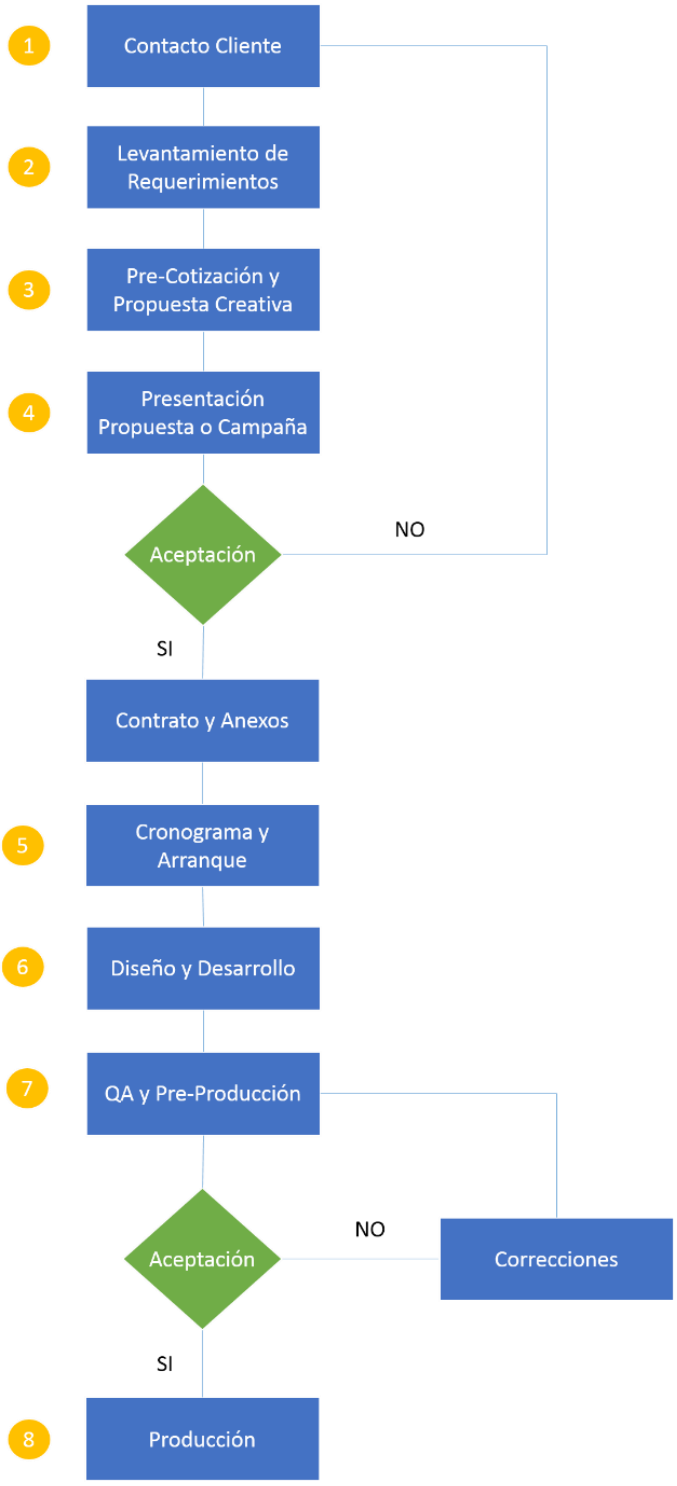
YAGE actualmente es una empresa con procesos ya definidos, sin embargo no existe una documentación clara y estos procesos no están estipulados formalmente. La empresa actualmente tiene los siguientes procesos relevantes a la alineación e implementación de ITIL® a sus procesos:

1. El CEO realiza el contacto principal, la venta, licitación o el cliente contacta a la empresa directamente.
2. Primera reunión con el cliente donde se analizan los objetivos que desea, aquí participan creatividad y cuentas.
    - Director Creativo y Ejecutivo de Cuentas encargado del cliente
3. Una vez claros los objetivos del cliente, cuentas realiza presupuestos y tiempos de entrega, creatividad define la campaña.
    - Se realiza una pre-cotización con los costos estimados de la campaña y/o desarrollo.
4. Se realiza una reunión con el cliente para presentar la campaña y una vez aprobada esta entra a producción.
    - Se realiza una cotización formal y un borrador de cronograma para que el cliente revise y acepte la propuesta.
    - Si se acepta la propuesta: se realiza un contrato, por lo general cubre lo que es la entrega de todo el proyecto, objetivos y fechas.
    - Un porcentaje del pago del contrato es en base al cumplimento de objetivos, si se cumple con los entregables se paga según el cumplimiento, caso contrario se hace un alcance según el porcentaje que se cumplió.
    - Se puede incluir en el contrato un “fee” por mantenimiento ya sea manejo de contenido, actualización, o manejo de redes sociales.

- Se puede agregar un anexo de acuerdo de servicio en caso de ser necesario donde se aclaren los temas de respuesta, soporte, cambios y aclaraciones de nuevas campañas, no es obligatorio y se lo hace muy pocas veces.

5. La productora realiza cronograma interno, hay la reunión de arranque con el equipo que va a participar.
    - Aquí participan todos los involucrados diseñadores, desarrolladores, ejecutivos de cuenta y otros que se considere necesario.
6. Empieza la producción en diseño, luego pasa a desarrollo y redes sociales dependiendo del caso y del proyecto.
7. Pruebas de funcionamiento y salida a pre-producción
8. Si todo va bien salida a producción, caso contrario, correcciones y salida a producción

Ilustración 4 - Proceso de Gestión de Proyectos



Fuente: YAGE
Elaborado por: Esteban Villacís y Santiago García

Una vez que entendimos cual es el proceso de la empresa nuestra propuesta se va a dirigir al desarrollo de una aplicación que permita a la empresa a tener una documentación organizada de su proceso así como también alinear su proceso al marco de trabajo de ITIL® V3 en los aspectos dentro del alcance de esta disertación.

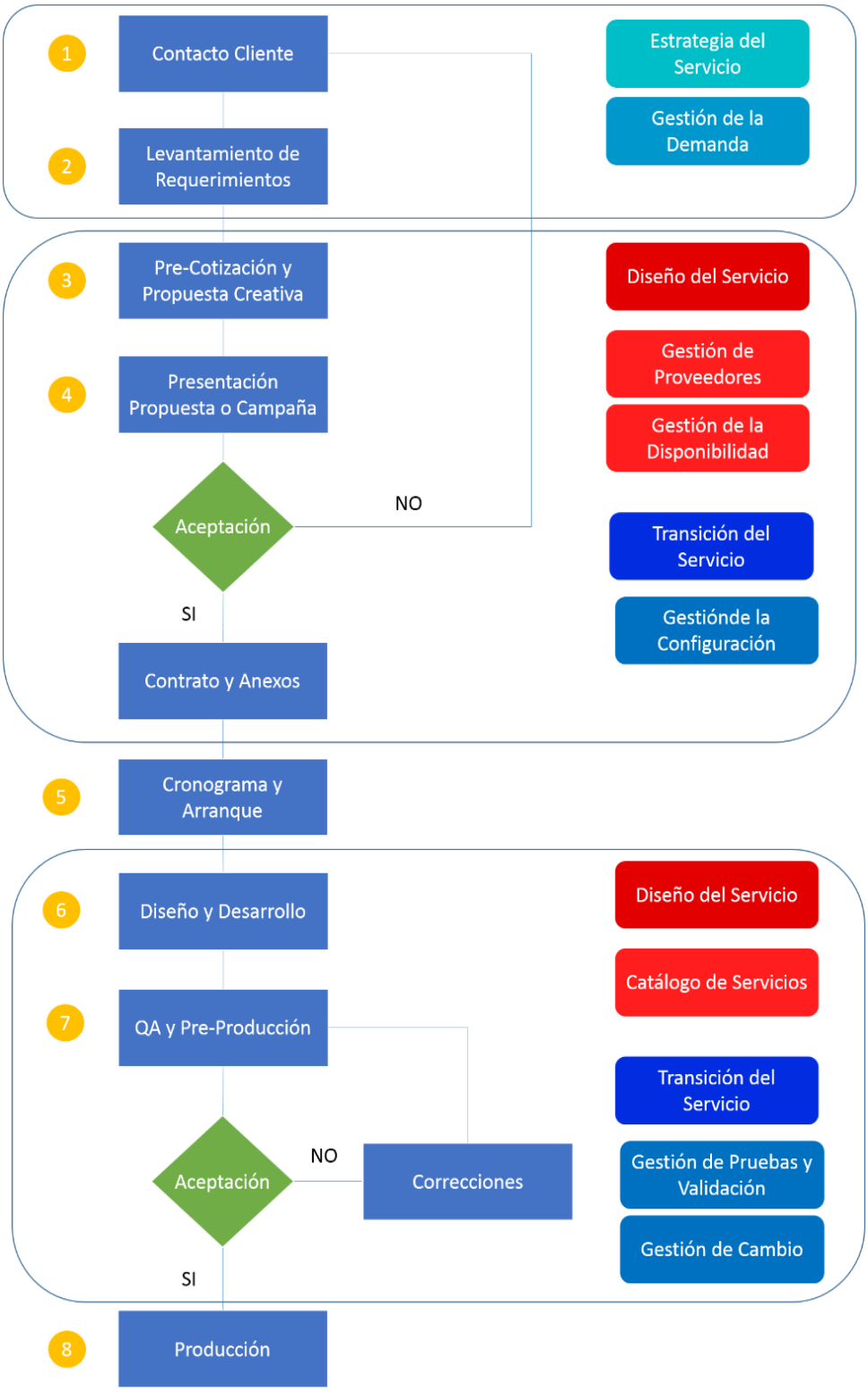
Para esto hemos desarrollado un diagrama explicativo de donde se encuentran los puntos a ser considerados dentro del proceso de la empresa:

- En el punto 1 se ingresa el cliente a la BDD de clientes para futuros contactos y referencias, todo esto se lo hace alineándonos a la Estrategia del Servicio que propone ITIL® V3, en este caso el proceso de ITIL® que se pretende automatizar y adaptar es la Gestión de la Demanda, que en conjunto con el Diseño del servicio se encargan de armonizar la respuesta de la empresa a la demanda.
- Entre los puntos 2 y 3 del proceso de la empresa hemos observado que en muchos casos es necesario contratar servicios externos por lo que se va a realizar una BDD de proveedores en conjunto con una administración del Servicio de Proveedores, todo esto alineado al Diseño del Servicio de TI donde se plantea la importancia de una gestión de proveedores. Dentro de esto se pretende tener un lugar donde subir documentación como OLAs[9] o contratos realizados por el proveedor además de contacto y condiciones básicas de su servicio.
- En el punto 4 se propone tener una base de datos de los servicios que presta la empresa así como también un repositorio de esta documentación y un formulario de acuerdo de servicio que permita mostrar la criticidad de cada uno de los servicios. Esta propuesta es en base a la recomendación del marco de trabajo de ITIL® que nos demuestra la importancia de tener una clara documentación del servicio que se da a los clientes y usuarios. Para esto nuestra propuesta es alinear el desarrollo al Diseño del Servicio, concretamente a los procesos de Gestión del Catálogo de Servicios y a la Gestión del Nivel de Servicio.
- En el punto 7 se encuentra la parte principal de la herramienta que se propone en esta disertación que es la gestión de errores y bugs en proyectos de desarrollo donde esta

[9] OLA (**operational-level agreement**) define las relaciones interdependientes para dar soporte a los acuerdos de nivel de servicio (SLA). El acuerdo describe las responsabilidades de cada grupo de soporte interno en relación a otros grupos de soporte, incluyendo los procesos y los tiempos en los que se entregan los servicios.

gestión se basa en los datos anteriormente ingresados dentro del proceso de gestión de proyectos de la empresa. La propuesta dentro de esta disertación es desarrollar una herramienta que se alinee con la Transición del Servicio propuesta en ITIL® donde vamos a atacar 2 procesos claves que son la Gestión de Validación y Pruebas y la Gestión de Cambios, sin embargo es muy importante aclarar que no se pretende automatizar completamente un sistema de gestión de cambios sino una manera de hacer un seguimiento a los cambios por correcciones en los sistemas desarrollados por la empresa.

Ilustración 5 - Propuesta de Mejora

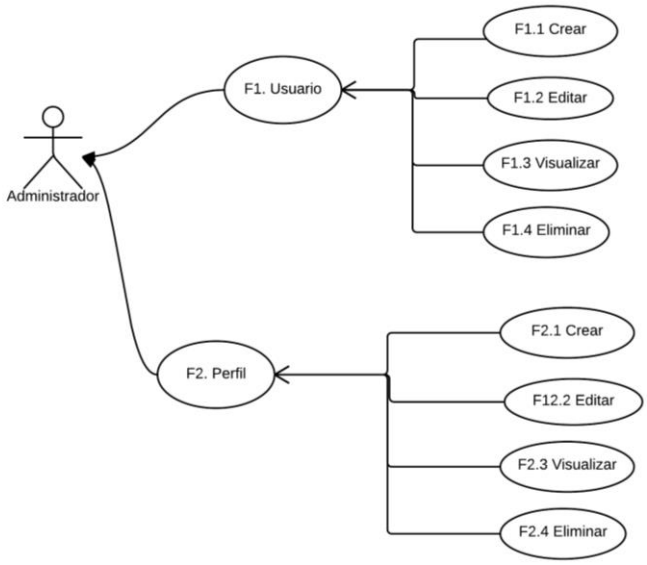


Fuente: YAGE
Elaborado por: Esteban Villacís y Santiago García

3.2 Estructura de Requerimientos

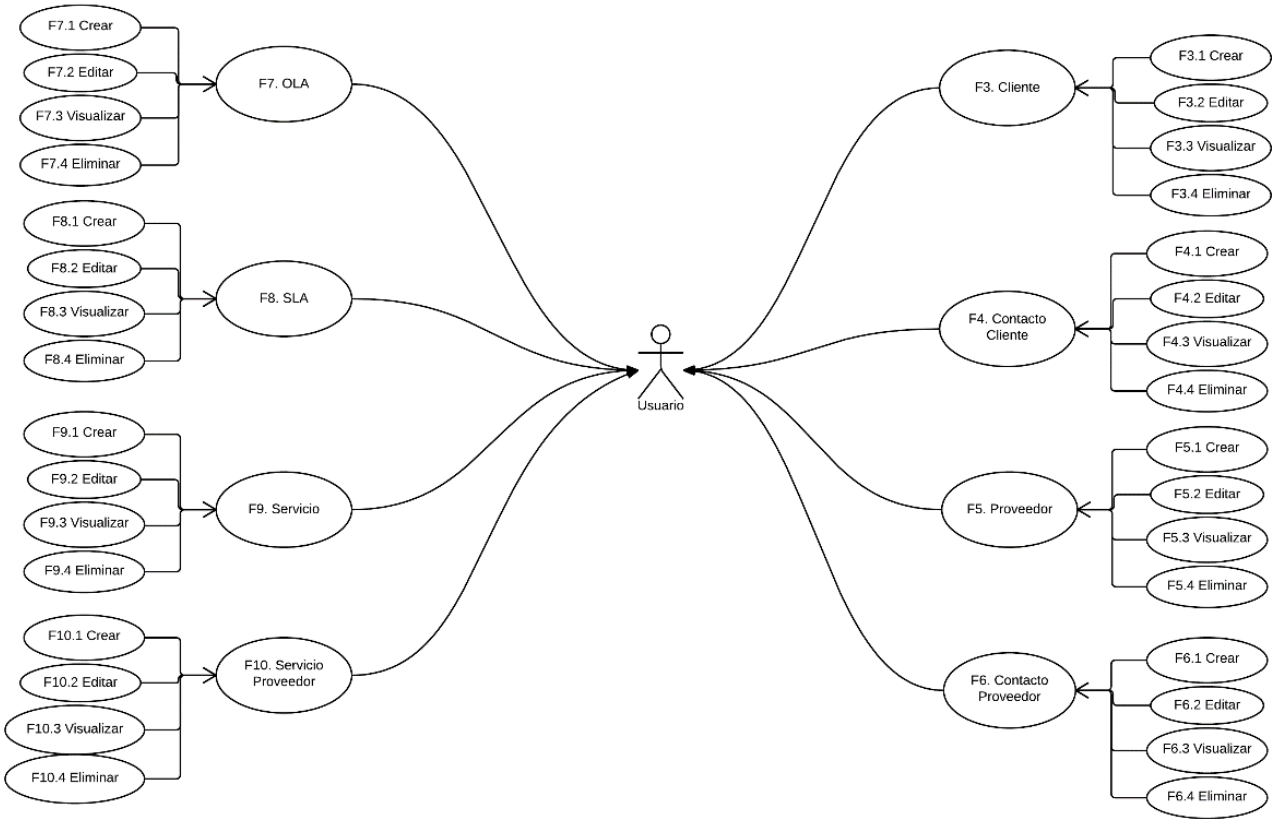
3.2.1 Diagrama de Casos de Uso

Ilustración 6 - Administración del Sistema



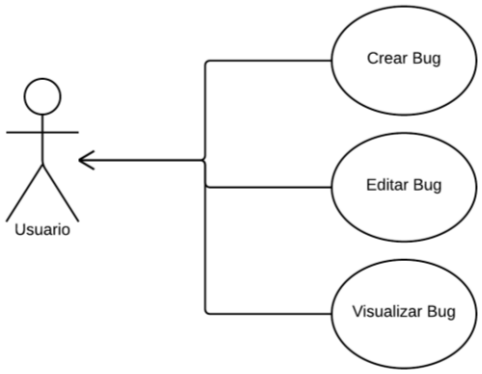
Fuente: Esteban Villacis, Santiago García
Elaborado por: Esteban Villacis, Santiago García

Ilustración 7 - Administración de Datos del Sistema



Fuente: Esteban Villacis, Santiago García
Elaborado por: Esteban Villacis, Santiago García

Ilustración8 - Administración de BUG's



Fuente: Esteban Villacis, Santiago García
Elaborado por: Esteban Villacis, Santiago García

3.2.2 Diagrama E-R

(Ver Anexo A)

3.3 Diseño

3.3.1 Diagrama de Procesos

(Ver Anexo B)

3.3.2 Diagramas de Secuencia

(Ver Anexo C)

3.3.3 Prototipo Diseño

Ilustración 9 - Login

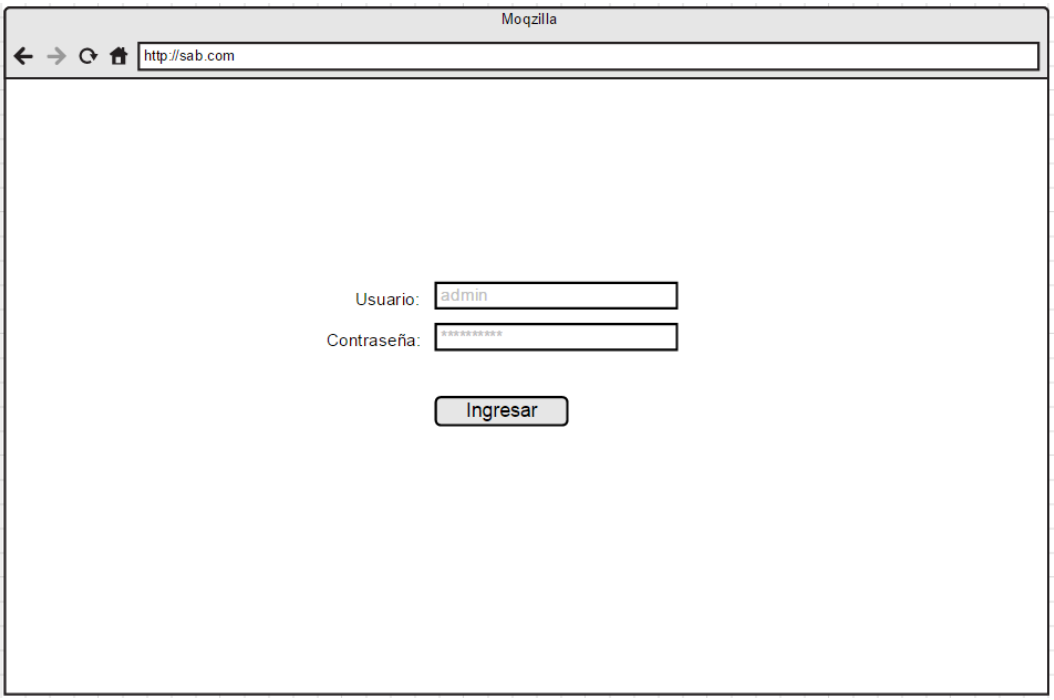
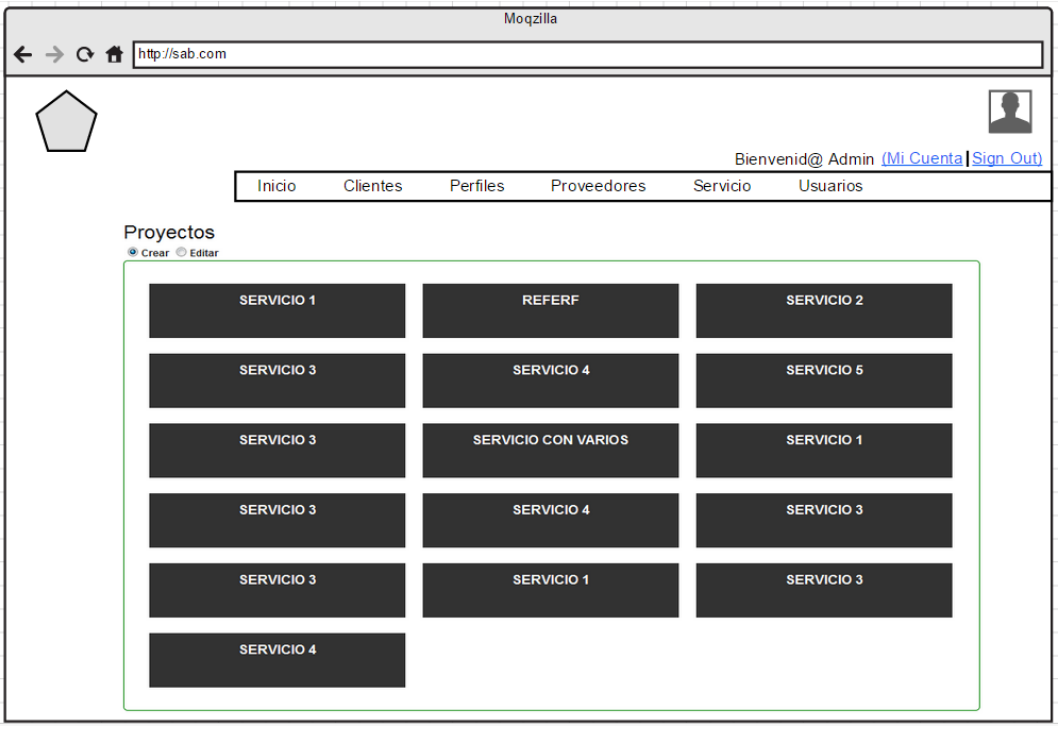


Ilustración10 - Página Principal



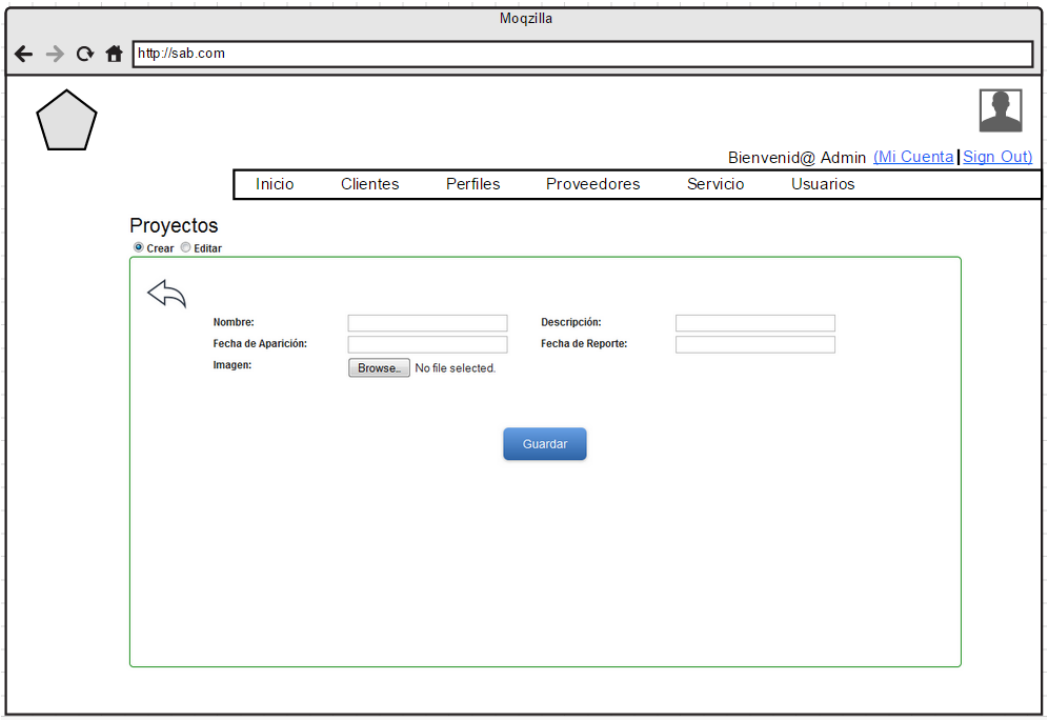
La pantalla principal muestra todos los proyectos o servicios que se han agregado anteriormente dentro de las administraciones del sistema. Es importante tener en cuenta que el sistema necesita que se creen los servicios para poder gestionar los errores dentro de cada uno de ellos y al momento de la creación de servicios es imperativo que los usuarios llenen los datos correspondientes al servicio, estos datos están apegados a las buenas prácticas mencionadas a lo largo de esta disertación.

Al momento de la creación de un servicio es necesario tener un cliente relacionado y los contactos relevantes en caso de existir, así mismo si el servicio o proyecto requiere de alguna contratación externa es necesario ingresar el proveedor correspondiente y los contactos en caso de ser necesarios, así mismo es muy importante recordar que las buenas prácticas nos hablan de tener un soporte, en este caso un OLA, que es el documento que respalda la entrega del servicio del proveedor.

Finalmente la parte más importante es el acuerdo de servicio que se realiza con el cliente y para esto es necesario llenar los datos correspondientes al acuerdo de servicio dentro del sistema como son el SLA (Acuerdo de Nivel de Servicio), plan de respaldos en caso de ser necesario que se apega a la recomendación de ITIL® de la gestión de la disponibilidad, todos estos datos se encuentran en la pantalla de creación del SLA que son

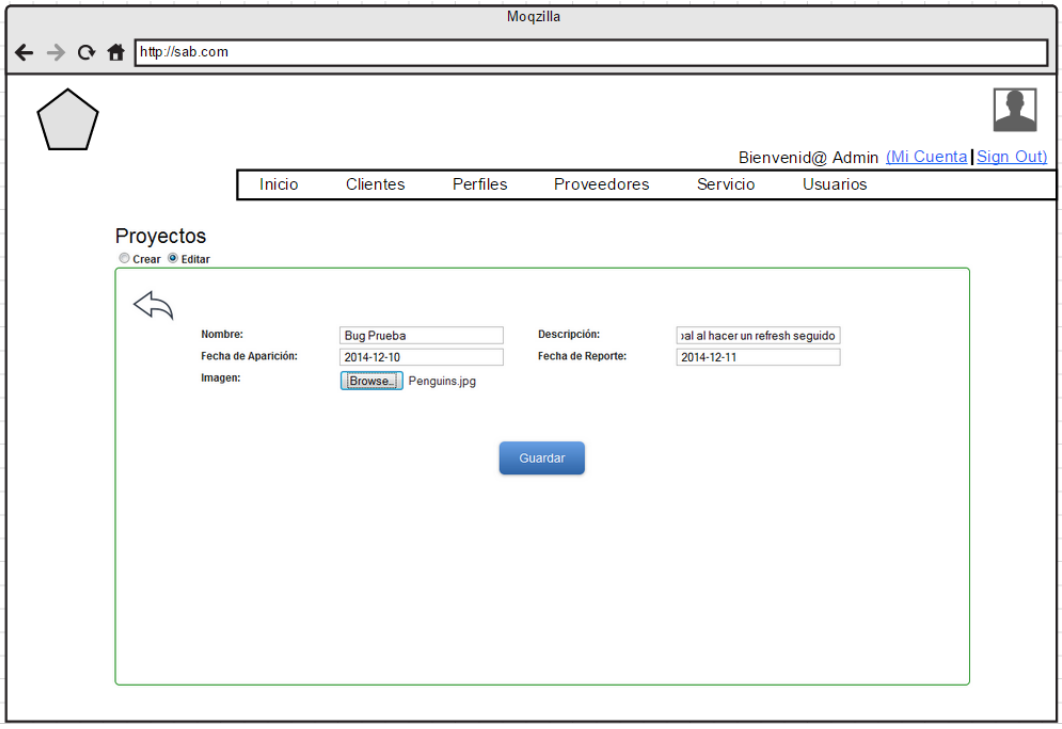
los que permitirán gestionar el servicio. No está de más mencionar que en el alcance de esta disertación no está incluido una administración de alertas para los acuerdos de servicio y esto queda para el proceso de la compañía, caso de estudio, para que se incluya una gestión diaria o semanal de los servicios que provee junto con la reunión de gestión de proyectos que se realiza a diario en este tipo de empresas.

Ilustración11 - Registrar Bug



Para el registro de un Nuevo Bug elegimos el botón radio correspondiente que en este caso es crear, una vez seleccionado crear elegimos cualquiera de los proyectos o servicios que se presentan en la pantalla principal del sistema, esto nos desplegará el formulario de un nuevo bug.

Ilustración12 - Editar Bug



Para el editar un Bug existente elegimos el botón radio correspondiente que en este caso es editar, una vez seleccionado editar elegimos cualquiera de los proyectos o servicios que se presentan en la pantalla principal del sistema, esto nos desplegará el formulario de edición de bugs.

Ilustración13 – Otras Administraciones CRUD

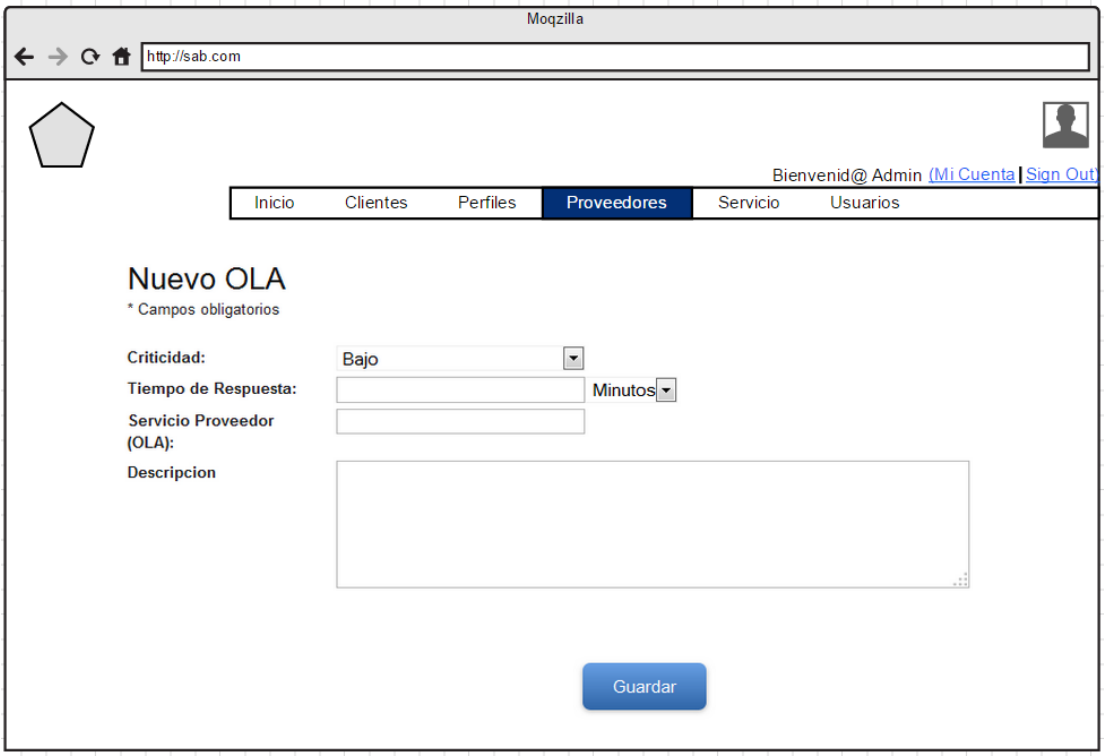


Ilustración14 – Otras Administraciones CRUD

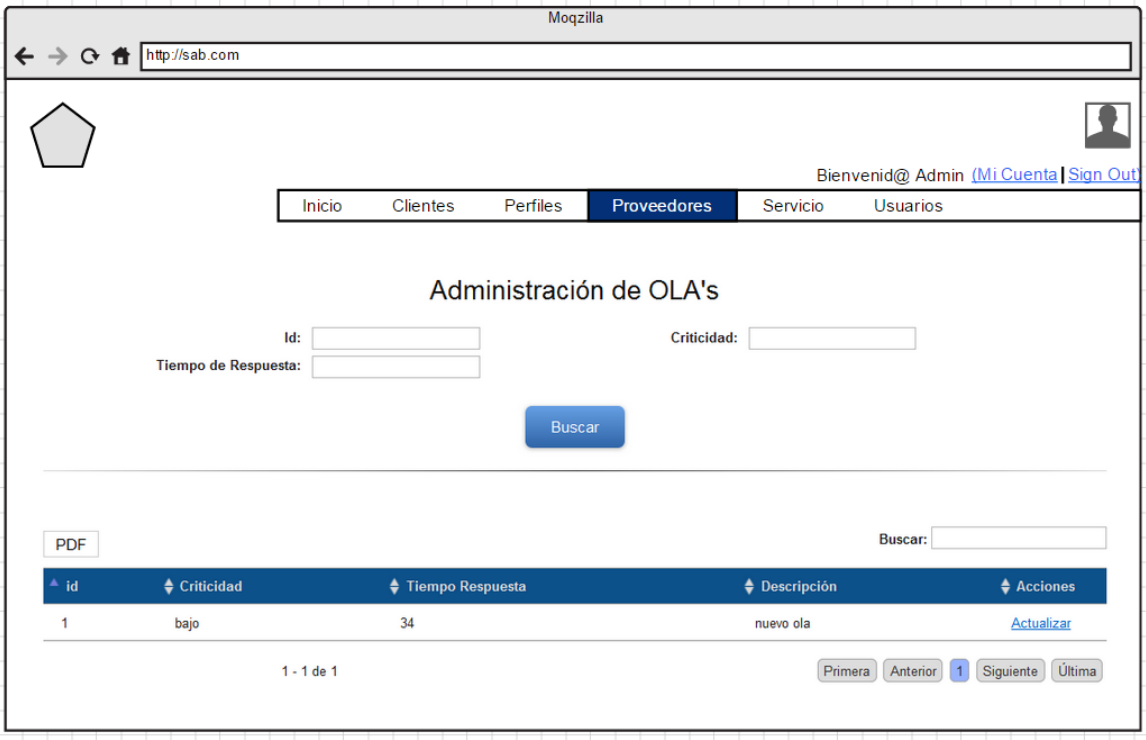
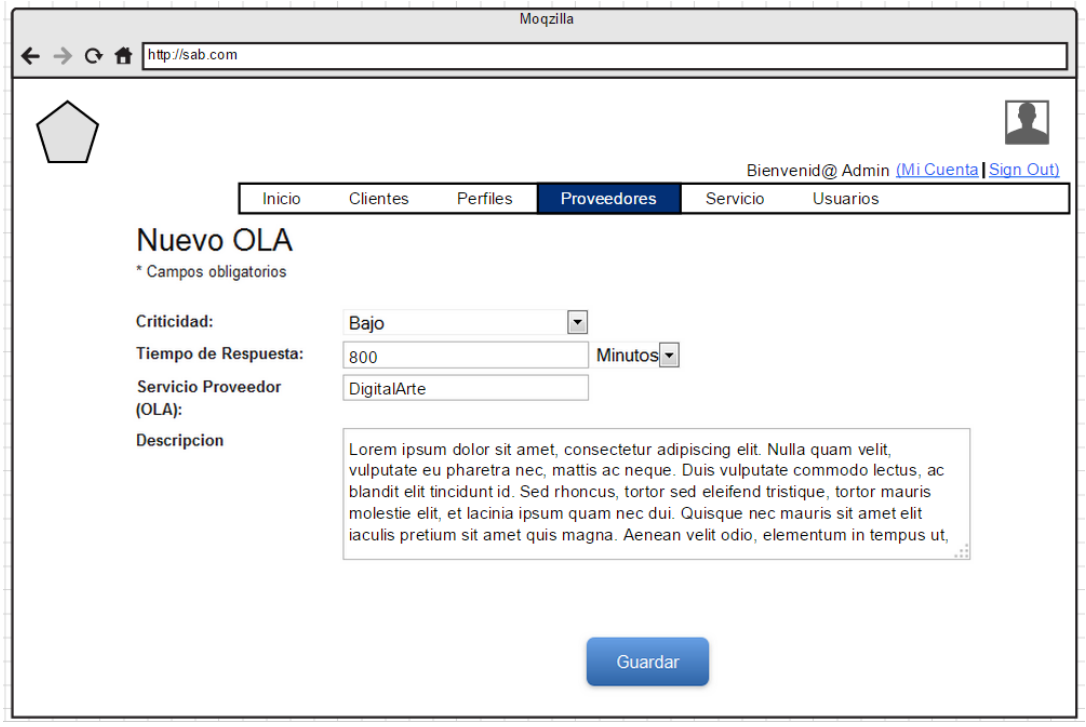


Ilustración15 – Otras Administraciones CRUD



Fuente: Santiago García y Esteban Villacís
Elaborado por: Santiago García y Esteban Villacís

CAPÍTULO IV: IMPLEMENTACIÓN Y PRUEBAS

4.1 Pruebas e Implementación

4.1.1 Estándares de Codificación

El emplear un estándar ayuda a la legibilidad del código por otra persona, esto es indispensable ya que estudios muestran que el 80% de programadores mantienen código implementado por otra persona, es decir es deseable mostrar orden y convenciones en la forma de programar el software.

- Se utilizará el esquema MVC[10], Modelo – Vista – Controlador para lo que se tendrán directorios separados para cada uno de estos componentes dentro de la organización del código desarrollado.
- Se ha definido la inserción de texto en Unicode para abarcar el mayor número de caracteres de todos los idiomas, en especial el español.
- Se ha restringido el uso del carácter "ñ" para nombres de cualquier tipo de variable, modelo, vista, controlador o archivo dentro de la codificación.
- Se ha definido que cualquier palabra que se encuentre dentro del código debe estar sin tilde para evitar problemas en la implementación.
- A la vez se pretende bautizar a cualquier variable, modelo, vista o controlador con un nombre que tenga referencia directa con su utilidad o lo que esta represente.
- Se ha tomado como estándar de programación el CamelCase.

  Este es el estándar de codificación al definir métodos, variables o palabras compuestas a lo largo del sistema. Lleva ese nombre ya que la construcción de la palabra compuesta asemeja las jorobas de un camello por el uso de mayúsculas al inicio de una nueva palabra. Este estándar se divide en dos grupos los cuales son el UpperCamelCase y el LowerCamelCase.

[10] MVC: El modelo–vista–controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

En la presente implementación se han ocupado los dos grupos definidos.

Para variables o nombres de métodos, se emplea el LowerCamelCase es decir la primera letra de la primera palabra empieza con minúscula y la primera letra de las palabras siguientes llevara mayúscula. Mientras que para nombres de los Modelos[11] se ha empleado UpperCamelCase, que indica que la primera letra de la primera palabra lleva mayúscula y la primera letra de las siguientes palabras también se escribirá con mayúsculas.

4.1.2 Implementación del Sistema

Para la implementación del Sistema utilizamos varias iteraciones que son las presentadas en el cuadro a continuación.

Tabla 1
Iteraciones de la Implementación del Sistema

Iteración 1
Configuración de Framework
Estándares para Codificación
Administración Clientes
Administración Proveedores
Iteración 2
Administración OLA
Administración Contacto Cliente
Administración Contacto Proveedor
Administración Contacto OLA
Administración Servicio
Administración SLA[12]
Iteración 3
Administración Usuarios
Administración Roles
Iteración 4
Menús y estilos
Despliegue de servicios
Administración Bugs
Crear Bugs
Editar Bugs

[11] Modelo: Dentro del esquema MVC, es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones.
[12] SLA (Acuerdo de Nivel de Servicio - ANS), es un contrato escrito entre un proveedor de servicio y su cliente con objeto de fijar el nivel acordado para la calidad de dicho servicio. El SLA o ANS es una herramienta que ayuda a ambas partes a llegar a un consenso en términos del nivel de calidad del servicio, en aspectos tales como tiempo de respuesta, disponibilidad, etc.

4.1.2.1 Código Fuente

El código fuente del sistema lo podemos encontrar en el siguiente enlace: https://github.com/sgarciab/SAB, así mismo lo podemos encontrar en el (*Anexo D)*.

4.1.3 Pruebas de Usuario

Las pruebas de validación, pruebas de usuario o pruebas de aceptación de usuario en la ingeniería de software son el proceso de revisión que verifica que el sistema de software producido cumple con las especificaciones requeridas y cumple con su objetivo. La validación es el proceso de comprobar que lo que se ha especificado es lo que el usuario realmente quería. Se trata de evaluar el sistema o parte de este durante o al final del desarrollo para determinar si satisface los requisitos iniciales.

En esta disertación solamente se realizaron pruebas de aceptación de usuario (UAT) donde la empresa fue la encargada de verificar la funcionalidad del sistema, dentro de estas pruebas se diseñaron scripts de pruebas con todos los casos críticos a ser probados por YAGE.

Tabla 2
UAT – INGRESO AL SISTEMA

UAT				
Nombre de prueba	Ingreso al Sistema			
Verificación	▪ Prueba de ingreso con varios usuarios ▪ Mensajes de advertencia ▪ Credenciales son correctas			
Categoría	Administración de Usuarios (F1.1)			
Prerrequisito	Usuario creado con permisos adecuados, conexión a BDD			
Prueba	**Pantalla**	**Acción**	**Resultado Esperado**	**Resultado Obtenido**
1	Ingreso al sistema	No ingresar datos e intentar ingresar al sistema	Se presenta el mensaje correspondiente	Mensaje desplegado es el esperado
2	Ingreso al sistema	Ingresar los datos incompletos para el acceso al Sistema SAB	Se presenta el mensaje correspondiente	Mensaje desplegado es el esperado
3	Ingreso al sistema	Ingresar los datos completos para el acceso al Sistema SAB	Se ingresa al sistema correctamente	Se ingresa satisfactoriamente

Fuente: Esteban Villacís y Santiago García
Elaborado por: Esteban Villacís y Santiago García

Tabla3
UAT – CREAR

UAT				
Nombre de prueba	Crear un Nuevo registro			
Verificación	▪ Prueba ingreso a una o varias administraciones para crear un nuevo registro ▪ Mensajes de advertencia ▪ Datos se graban correctamente en la Base de Datos			
Categoría	CrearCliente (F3.1)			
Prerrequisito	- Ingreso al sistema correctamente - Usuario creado con los permisos correspondientes			
Prueba	**Pantalla**	**Acción**	**Resultado Esperado**	**Resultado Obtenido**
1	Ingreso al sistema	Ingreso al sistema.	Se ingresa al sistema correctamente	Se ingresa al sistema correctamente
2	CrearClientes	Ingresar los datos y grabar.	Se guarda el registro correctamente	Se guarda un nuevo registro
3	Mensajes de error	Ingresar los datos incompletos y tratar de guardar.	Muestra los mensajes de error por datos incompleto No debe dejar grabar	Se muestran los mensajes satisfactoriamente No se puede guardar con datos incompletos
4	Mensajes de creación	Se guarda un registro y un mensaje de creación de nuevo registro es desplegado.	Mensaje de nuevo registro guardado se muestra satisfactoriamente	Se muestra el mensaje correctamente

Fuente: Esteban Villacís y Santiago García
Elaborado por: Esteban Villacís y Santiago García

Tabla4
UAT – EDITAR

UAT				
Nombre de prueba	Editar un registro			
Verificación	▪ Prueba editar a una o varias administraciones para editar un registro ▪ Mensajes de advertencia ▪ Datos se graban correctamente en la Base de Datos			
Categoría	EditarCliente (F3.2)			
Prerrequisito	- Ingreso al sistema correctamente - Usuario creado con los permisos correspondientes			
Prueba	**Pantalla**	**Acción**	**Resultado Esperado**	**Resultado Obtenido**
1	Ingreso al sistema	Ingreso al sistema.	Se ingresa al sistema correctamente	Se ingresa al sistema correctamente
2	Ingreso a Edición	Ingresar a Búsqueda y Despliegue de registros, ingresamos a editar uno de los registros ya creados	Se ingresa correctamente	Como el esperado
3	EditarClientes	Ingresar los datos y grabar.	Se guarda el registro correctamente	Se guarda el registro
4	Mensajes de error	Ingresar los datos incompletos y tratar de guardar.	Muestra los mensajes de error por datos incompleto No debe dejar grabar	Se muestran los mensajes satisfactoriamente No se puede guardar con datos incompletos
5	Mensajes de creación	Se guarda un registro y un mensaje de edición de un registro es desplegado.	Mensaje de edición de un registro guardado se muestra satisfactoriamente	Se muestra el mensaje correctamente

Fuente: Esteban Villacís y Santiago García
Elaborado por: Esteban Villacís y Santiago García

Tabla 5
UAT – MOSTRAR Y BUSCAR

UAT				
Nombre de prueba	Mostrar y Buscar registros			
Verificación	▪ Prueba búsqueda y despliegue de uno o varios registros ▪ Mensajes de advertencia			
Categoría	CrearCliente (F3.1)			
Prerrequisito	- Ingreso al sistema correctamente - Usuario creado con los permisos correspondientes			
Prueba	**Pantalla**	**Acción**	**Resultado Esperado**	**Resultado Obtenido**
1	Ingreso al sistema	Ingreso al sistema.	Se ingresa al sistema correctamente	Se ingresa al sistema correctamente
2	BuscarClientes	Ingresar los datos y buscar.	Se muestra el registro buscado o no muestra nada si no existen registros que concuerden con la búsqueda	Como el esperado
3	MostrarClientes	Buscar todos los registros.	Muestra un grid con todos los registros	Se muestra el grid correctamente
4	Exportar PDF	Se exportan todos los datos a PDF	Se exportan los datos correctamente	Como el esperado

Fuente: Esteban Villacís y Santiago García
Elaborado por: Esteban Villacís y Santiago García

Tabla 6
UAT – ADMINISTRACIÓN DE BUGS

UAT				
Nombre de prueba	Ingresar y Editar BUG's de Servicios			
Verificación	▪ Prueba creación, edición, búsqueda y despliegue de uno o varios bugs ▪ Mensajes de advertencia, y otros ▪ Imágenes correspondientes a los bugs se graban y se editan correctamente			
Categoría	Administración BUG's			
Prerrequisito	- Ingreso al sistema correctamente - Usuario creado con los permisos correspondientes - Datos de clientes, responsables, OLAs, SLAs, etc se han agregado anteriormente - Datos de servicios es indispensable para esta prueba			
Prueba	**Pantalla**	**Acción**	**Resultado Esperado**	**Resultado Obtenido**
1	Ingreso al sistema	Ingreso al sistema.	Se ingresa al sistema correctamente	Se ingresa al sistema correctamente
2	Selección de Servicios	Seleccionar el servicio correspondiente	Se selecciona el servicio según sea nuevo bug o edición de uno existente	Como el esperado
3	Nuevo Bug	Crear un nuevo Bug	Se muestra el formulario de creación y se guarda correctamente	Como el esperado
4	Editar Bug	Se ingresa al servicio correspondiente y se elige el bug a ser editado	Se muestran y editan los datos correctamente	Como el esperado
5	Mensajes e ingreso de imágenes de Bugs	Se muestran los mensajes de error satisfactoriamente Se graba y edita la imagen correctamente	Se muestran los mensajes de error satisfactoriamente Se graba y edita la imagen correctamente	Como el esperado

Fuente: Esteban Villacís y Santiago García
Elaborado por: Esteban Villacís y Santiago García

4.1.6 Impacto Funcional del Sistema de Administración de Bugs

El Sistema de Administración de Bugs y Errores ayuda a la empresa a tener una base de datos de proveedores, clientes y servicios así mismo como un seguimiento de cambios en errores de los servicios que provee la empresa. Ninguna de estas cosas mencionadas

anteriormente se las tomaba en cuenta dentro de los procesos de la compañía pero estaban ahí, por lo que ahora se puede tener un seguimiento más sencillo de los clientes y de los proyectos que estos tienen con YAGE, también al tener una base de datos de proveedores baja la incidencia de problemas al tener siempre un contrato de servicio claro.

Adicionalmente se tiene una herramienta de seguimiento de bugs y errores que permite hacer de alguna manera un seguimiento de la eficiencia del grupo de desarrollo de la empresa.

Una vez que se utilice la herramienta se podrán recopilar datos más reales del impacto real del sistema.

4.1.5 Mantenimiento del Sistema

Aunque se haya definido de una manera correcta el plan de disponibilidad, existirán ciertas interrupciones no previstas anteriormente. En esos casos es necesario recuperar el servicio de la manera más rápida, evitando un gran impacto en la calidad del servicio. La Gestión de la Disponibilidad asesora a la Gestión de Incidencias para que exista un plan de recuperación tomando en cuenta las necesidades de disponibilidad del negocio y los procesos necesarios para restaurar el servicio.

Hemos acordado mantener un repositorio común público para la recuperación del sistema en caso de ser necesario, en conjunto con esté repositorio se incluyen las instrucciones y los manuales respectivos para la configuración y operación del sistema.

Manual Técnico:

(Ver Anexo E)

Manual de Usuario:

(Ver Anexo F)

4.2 Benchmark otras soluciones de corrección de errores

(Ver Anexo G)

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- La falta de organización en los procesos internos es uno de los principales factores a mejorar a la hora de optimizar los recursos que se utilizan en la gestión del servicio, es por esto que es indispensable que se tenga un mapa de procesos claro donde se puedan identificar los actores y las tareas que se realizan en una empresa sin importar su tamaño.

- El marco de trabajo ITIL®, contribuye a la integración entre TI y el negocio, convirtiéndose en un pilar importante en la superación y éxito de la organización, sin embargo hemos visto que el marco de trabajo de ITIL® debe ser adaptado para organizaciones pequeñas y medianas ya que la carga de trabajo de los procesos de ITIL® genera problemas por la falta de los recursos que propone el marco de trabajo de ITIL®.

- En esta disertación elegimos una metodología ágil de desarrollo, XP o Extreme Programming es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Elegimos esta metodología con el objetivo de compartir conocimiento y desarrollar esta disertación en el menor tiempo posible fomentando nuestro aprendizaje técnico e intrapersonal.

- Es indispensable un proceso de aseguramiento de la calidad del software dentro de la empresas de desarrollo de software mediante cualquier metodología de pruebas que asegure que el producto cumple con las expectativas del cliente, es por esto que nuestro producto se convierte en un activo que puede cumplir con la tarea de asegurar la calidad de los servicios que provee la empresa mediante la gestión de bugs y errores generados en proyectos de desarrollo de software.

- En todos los proyectos tecnológicos es necesario escoger bien las herramientas de desarrollo, tanto como el stack tecnológico[13], de esto dependerá la curva de aprendizaje para los desarrolladores, gastos en capacitación y precios de mantenimiento. Esto depende mucho del giro de negocio de la empresa y cuál es su enfoque de crecimiento tecnológico.

- Las empresas como en este caso YAGE nos demuestra que existe una demanda en el mercado de herramientas como la desarrollada en esta disertación ya que además de automatizar ciertos procesos y buenas prácticas de ITIL® se identificaron y modelaron los procesos que se encontraban informalmente documentados lo cual ha sido de ayuda para la empresa y al mismo tiempo nos demuestra que este tipo de empresas necesitan trabajar en la formalización de sus procesos para generar proyectos, servicios y sistemas de mejor calidad.

RECOMENDACIONES

- Al momento de tener poca o ninguna documentación de los servicios que provee una empresa de desarrollo de software se crea una dependencia de recursos lo que puede generar problemas a futuro, es por esto que recomendamos tener políticas claras donde se documentación de los procesos que realiza cada persona dentro de una compañía.

- Hay una necesidad de la empresa de organización de sus recursos y de sus procesos, es por lo que recomendamos que se implemente el marco de trabajo de ITIL® formalmente dentro de la empresa, que como hemos mencionado no necesariamente se deben aplicar todos los procesos de ITIL® sino aplicar los necesarios para que la empresa se beneficie.

[13] [9] Conjunto de tecnologías que trabajan en conjunto y son complementarias, ejemplo XAMPP= Apache, MySQL, PHP, Perl. MEAN= MongoDB - Express - AngularJS - Node.JS

- El ciclo de vida de un proyecto de desarrollo de software es muy importante, e indistintamente de cual metodología de desarrollo se escoja recomendamos que se ponga mucho énfasis en la fase de planificación, que para nosotros es la más importante, ya que aquí se puede ver el alcance real del proyecto y dependiendo de esto se podrá escoger las mejores herramientas y el stack tecnológico.

- La gestión de errores y bugs en proyectos de desarrollo de software es indispensable por lo que recomendamos hacer uso de la herramienta generada en esta disertación y a futuro poner más énfasis en la gestión del aseguramiento y mejora de la calidad, en el desarrollo de software, como un proceso indispensable dentro del ciclo de vida de un sistema.

- Dado el alcance de esta disertación consideramos que existen procesos críticos que se podrían incluir en futuras implementaciones y mejoras al sistema como son la gestión de alertas tanto para los acuerdos de servicio con los proveedores como con los clientes y los datos propios de estos como la disponibilidad y la criticidad de cada uno de estos, así mismo una administración de alertas para los bugs encontrados según se requiera en el lugar donde se implemente este sistema.

REFERENCIAS

Libros y Publicaciones

Arraj, V. (2013). ITIL®: The Basics.

Cartlidge, A., Rudd, C., Smith, M., Wigzel, P., Rance, S., Shaw, S., & Wright, T. (2012). An Introductory Overview of ITIL® 2011. Berkshire, United Kingdom.

Great Britain. Cabinet Office. (2011). *ITIL® Service Design.* The Stationery Office.

Great Britain. Cabinet Office. (2011). *ITIL® Service Transition.* TSO.

Kniberg, H. (2007). Scrum & XP from the trenches. United States.

The Stationery Office. (2011). *ITIL® Continual Service Improvement.* London: The Stationery Office - TSO.

Fuentes Web

Osiatis, I. (s.f.). *Diseño para los servicios TI*. Recuperado el 28 de Diciembre de 2014, de ITIL v3 Gestión de servicios TI: http://itilv3.osiatis.es/diseno_servicios_TI.php

Osiatis, I. (s.f.). *El ciclo de vida de los servicios de TI*. Recuperado el 28 de Diciembre de 2014, de ITIL v3 Gestion de servicios TI: http://itilv3.osiatis.es/ciclo_vida_servicios_TI.php

Osiatis, I. (s.f.). *Estrategia de los servicios de ITIL®*. Recuperado el 28 de Diciembre de 2014, de ITIL® v3 gestion de servicios TI: http://itilv3.osiatis.es/estrategia_servicios_TI.php

PHP.net. (s.f.). *PHP.net*. Obtenido de http://php.net/manual/es/intro-whatis.php

Process Map, I. (5 de Agosto de 2013). *ITIL para pequeñas y medianas empresas*. Recuperado el 28 de Diciembre de 2014, de IT Process Map: http://wiki.es.it-processmaps.com/index.php/ITIL_para_peque%C3%B1as_y_medianas_empresas

Universidad Union Simon Bolivar. (24 de 01 de 2015). Obtenido de http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html

Wells, D. (s.f.). *Extreme Programming*. Obtenido de http://www.extremeprogramming.org/map/project.html

Wikipedia. (s.f.). Obtenido de http://es.wikipedia.org/wiki/NetBeans

Wikipedia. (s.f.). Obtenido de http://es.wikipedia.org/wiki/MySQL

Wikipedia. (s.f.). Obtenido de http://es.wikipedia.org/wiki/Kohana

wikipedia. (s.f.). *Wikipedia*. Obtenido de http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos#Caracter.C3.ADsticas_de_la_POO