Segana©

Plan de integración continua y pruebas de concepto

Version <1.1>

Historial de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 17/02/2012 | 1.0 | Primera versión del documento | Kibo S.A. |
| 17/02/2012 | 1.1 | Revision Detalles | Kibo S.A. |

Tabla de Contenidos

Contenido

[1. Introduccion 6](#_Toc317285363)

[1.1 Proposito 6](#_Toc317285364)

[1.2 Ambito de la Aplicacion 6](#_Toc317285365)

[1.3 Referencias 6](#_Toc317285366)

[1.4 Descripción general: 6](#_Toc317285367)

[2. Selección de herramientas: 7](#_Toc317285368)

[Integración continua 7](#_Toc317285369)

[Herramienta de Integración Continua 7](#_Toc317285370)

[Herramientas de control de versiones 8](#_Toc317285371)

[Evaluación de Herramientas de IC 9](#_Toc317285372)

[2.1 Aspectos a Evaluar 9](#_Toc317285373)

[2.2 Criterios Importantes 9](#_Toc317285374)

[2.3 Herramientas a evaluar 10](#_Toc317285375)

[2.4 Metodología de evaluación 10](#_Toc317285376)

[2.5 Hudson 11](#_Toc317285377)

[2.5.1 Opensource 11](#_Toc317285378)

[2.5.2 SoportedelLenguaje 11](#_Toc317285379)

[2.5.3 Precio 12](#_Toc317285380)

[2.5.4 Control de bugs 12](#_Toc317285381)

[2.5.5 Soporte para Herramienta de Control de Versiones 12](#_Toc317285382)

[2.5.6 SoporteparaPublicación 13](#_Toc317285383)

[2.5.7 Interfaz 13](#_Toc317285384)

[2.5.8 Herramientas de Build que soporta. 14](#_Toc317285385)

[2.5.9 Herramientas de Pruebas que soporta. 14](#_Toc317285386)

[2.5.10 Dependencias y plataformas. 15](#_Toc317285387)

[2.5.11 Grafica de Resultados 15](#_Toc317285388)

[2.6 CruiseControl 16](#_Toc317285389)

[2.6.1 Opensource 16](#_Toc317285390)

[2.6.2 SoportedelLenguaje 16](#_Toc317285391)

[2.6.3 Precio 16](#_Toc317285392)

[2.6.4 Control de bugs 16](#_Toc317285393)

[2.6.5 Soporte para Herramientas de control de versiones 17](#_Toc317285394)

[2.6.6 SoporteparaPublicaciones 17](#_Toc317285395)

[2.6.7 Interfaz 18](#_Toc317285396)

[2.6.8 Herramientas de Build que soporta. 18](#_Toc317285397)

[2.6.9 Herramientas de Pruebas que soporta. 18](#_Toc317285398)

[2.6.10 Dependencias y plataformas. 19](#_Toc317285399)

[2.6.11 Grafica de Resultados 19](#_Toc317285400)

[Evaluación de Herramientas de control de Versiones 20](#_Toc317285401)

[2.6.12 Aspectos a Evaluar 20](#_Toc317285402)

[2.6.13 Criterios Importantes 20](#_Toc317285403)

[2.7 Herramientas a evaluar 21](#_Toc317285404)

[2.8 Metodología de evaluación 21](#_Toc317285405)

[2.9 Subversion 22](#_Toc317285406)

[2.9.1 Facilidad de Uso 22](#_Toc317285407)

[2.9.2 Manejo de Branches 22](#_Toc317285408)

[2.9.3 Manejo de Merges 22](#_Toc317285409)

[2.9.4 Grafica 22](#_Toc317285410)

[2.10 Git 23](#_Toc317285411)

[2.10.1 Facilidad de Uso 23](#_Toc317285412)

[2.10.2 Manejo de Branches 23](#_Toc317285413)

[2.10.3 Manejo de Merges y commits 23](#_Toc317285414)

[2.10.4 Grafica 23](#_Toc317285415)

[Conclusiones 24](#_Toc317285416)

[2.11 Resultados de Herramienta CI 24](#_Toc317285417)

[2.11.1 Hudson 24](#_Toc317285418)

[2.11.2 CruiseControl 24](#_Toc317285419)

[2.12 Resultados de Herramientas de control de Versiones 25](#_Toc317285420)

[2.12.1 Subversion 25](#_Toc317285421)

[2.12.2 Git 25](#_Toc317285422)

[3. Descripción de la herramienta seleccionada 26](#_Toc317285423)

[3.1 Para desarrollo 26](#_Toc317285424)

[3.2 Herramienta para manejo de Versiones 26](#_Toc317285425)

[3.3 Herramienta para Manejar Builds y Dependencias 26](#_Toc317285426)

[3.4 Herramienta para Manejo de Integracion continua 26](#_Toc317285427)

[3.5 Funcionamiento 26](#_Toc317285428)

[4. Estructura de directorios 27](#_Toc317285429)

[4.1 Definiciones, AcrónimosyAbreviaciones 27](#_Toc317285430)

[4.2 Estructura de directorios 27](#_Toc317285431)

[5. Esquema de identificación de objetos 28](#_Toc317285432)

[5.1 Tipos de Archivos 28](#_Toc317285433)

[5.1.1 Archivos de configuración 28](#_Toc317285434)

[Estos archivos se refieren a los bach que sean necesarios crear para el funcionamiento correcto de la aplicación. Estos pueden ser creados para la instalación o para la creación de algún directorio necesario. 28](#_Toc317285435)

[5.1.2 Ejecutables: 28](#_Toc317285436)

[5.1.3 Clases 28](#_Toc317285437)

[5.1.4 Paginas 28](#_Toc317285438)

[5.1.5 Base de datos 28](#_Toc317285439)

[5.2 Nombres 28](#_Toc317285440)

[5.2.1 Java: 28](#_Toc317285441)

[5.2.2 Jsp 29](#_Toc317285442)

[5.2.3 Bases de datos 29](#_Toc317285443)

[5.3 Estándares 30](#_Toc317285444)

[5.3.1 Estándares para la documentación: 30](#_Toc317285445)

[5.3.2 Metodología a utilizar 30](#_Toc317285446)

[6. Esquema de nomenclatura de versiones 31](#_Toc317285447)

[7. Proceso de liberación 32](#_Toc317285448)

[7.1 Responsabilidades 32](#_Toc317285449)

[7.1.1 Organigrama 32](#_Toc317285450)

[7.2 Programa (calendario, horario) 33](#_Toc317285451)

[7.3 Plan de costos 34](#_Toc317285452)

[7.3.1 Hardware 34](#_Toc317285453)

[7.3.2 Software 34](#_Toc317285454)

[(Se utilizaran herramientas open source): 34](#_Toc317285455)

[7.3.3 Servicios 34](#_Toc317285456)

[7.3.4 Personal 34](#_Toc317285457)

[7.3.5 Capacitación 34](#_Toc317285458)

[7.3.6 Costo aproximado 35](#_Toc317285459)

[7.4 Capacitación 35](#_Toc317285460)

[7.4.1 Metodología de Capacitación 35](#_Toc317285461)

[8. Pruebas de concepto 38](#_Toc317285462)

Plan de integración continua y pruebas de concepto

# Introduccion

Este documento tiene como objetivo plantear en base a conocimientos técnicos la manera en la cual se desarrollara el sistema para apuestas por internet: Segana© el objetivo es que a través de este documento quede establecida una guía para los desarrolladores de que forma se realizara el sistema.

Experiencias previas nos indican que a lo largo del desarrollo se iran encontrando distintas barreras que dificultaran o harán lento el proceso de desarrollo por lo que se espera elegir de buena manera la herramienta y definir los estándares para que faciliten cualquier complejidad que exista.

## Proposito

Determinar los estándares de desarrollo bajo los cuales se realizara la aplicación, esto incluye todas las herramientas que vayamos a utilizar.

Así mismo se espera escoger las herramientas que serán utilizadas a través de un profundo análisis de varias herramientas hasta seleccionar la mejor.

Se deberá dejar los cimientos establecidos para la arquitectura que formara parte de la aplicación.

## Ambito de la Aplicacion

Toda la información descrita formara parte importante del desarrollo ya que proporcionara guías tanto como para el diseño como para el desarrollo del producto.

Selección de herramientas: Se determinaran las herramientas que nos acompañaran en el desarrollo.

Definición de la arquitectura: Se definirá la arquitectura que formara parte del sistema así mismo el diseño de los archivos y las guias para la comunicación entre los mismos.

Estandares: Se determinaran los estándares que determinaran los nombres y en algunos casos la estructura de los elementos que componen el sistema.

## Referencias

La información descrita en este documento servirá como base para la realización tanto de la documentación que será entregada en próximas entregas como de los objetos que formarán parte del desarrollo del sistema.

## Descripción general:

Este documento contiene información de vital importancia para el proceso de desarrollo del sistema Segana, el cual tiene como objetivo presentarles a los usuarios una interfaz rápida y amistosa, que tenga las mejores ofertas y las mejores opciones para el sistema de apuestas por internet de Guatemala.

Estos estándares definirán métricas de desarrollo que servirán como guía para los programadores, que serán quienes en mayor parte las utilizarán.

# Selección de herramientas:

Análisis de Herramienta de IC

# Integración continua

Para el desarrollo de software existe una variedad de metodologías de desarrollo que determina en gran parte lo que se va a hacer, como se va a hacer, cuando se va a entregar, como se va a trabajar entre otros aspectos y establece generalmente desde los artefactos generados a través de la implementación de la misma hasta la configuración de los grupos de trabajo, así como la manera de organizar y mantener el código del software junto con las responsabilidades que esto implica, la metodología a elegir depende mayormente del tipo de proyecto que se ejecute, el tamaño del mismo, el grupo de trabajo con el que se cuenta, el tiempo y la experiencia de quien lo lleva a cabo entre otros elementos, es una estructura de planificación, estructurar, desarrollar y dar seguimiento a un proyecto de software.

Integración continua es una metodología de desarrollo de software que se basa desde el punto de vista de desarrollo en realizar liberaciones o entregas de software de la aplicación o proyecto en el que se trabaja de manera constante en lapsos de tiempo muy cortos, donde se puede esperar ver varias liberaciones de código en un mismo día, una de las ventajas de trabajar de esta manera es que se cuenta con retroalimentación rápida del código que se libera sobre la aplicación, también se puede contar si el proceso de automatizado es el correcto con liberación directamente a las aplicaciones que el usuario utiliza contando este siempre con la última versión y los últimos cambios y por lo tanto un proceso de actualización más efectivo y transparente para el cliente, la integración continua es asociada con metodologías de desarrollo agil y extreme programming.

# Herramienta de Integración Continua

Desde el punto de vista técnico y de la configuración del entorno de trabajo, la integración continua requiere que se construya inicialmente un entorno de desarrollo que comprende procesos como compilación, construcción, despliegue y pruebas, para esto se necesita hacer uso de herramientas que manejen la compilación, el lenguaje de programación en que se desarrolla y se lleva a cabo a través de Trabajos ejecutados por una herramienta de integración continua y del resto de la arquitectura de la configuración del proyecto, se requiere de un repositorio en el caso que se trabaje con varios desarrolladores de manera simultánea y una herramienta de IC que tenga soporte para este repositorio, la herramienta también necesita hacer uso de alguna herramienta que genere builds y manejo de dependencias, herramientas como maven y ant, asi como herramientas de control de versiones para manejar el código releases y versiones del software.

# Herramientas de control de versiones

Las herramientas de control de versiones hacen más fácil la administración de las versiones de una aplicación o pieza de software y en muchos casos también de la documentación y la estructura de los archivos relacionados con el proyecto en cuestión, es decir se puede esperar encontrar en repositorio de versiones de software además del código fuente de la aplicación la documentación como casos de uso, diagramas de estado, entre otros artefactos generados por el equipo de trabajo.

Existen dos clasificaciones de herramientas de control de versiones, las centralizadas y las distribuidas, se espera que una herramienta de control de versiones provea de una forma de almacenar, que permita realizar cambios en los archivos almacenados y que cuente con un historial de los cambios realizados así como una explicación de cada cambio llevado a cabo, entre las herramientas conocidas en el mercado se encuentran CVS, Subversion, Git, Mercurial.

Entre las operaciones que se suelen llevar a cabo sobre un repositorio de versiones están checkout (que sirve para descargar los archivos de un repositorio) checkin, commit, submit (que sirven para subir archivos o actualizar archivos al repositorio), las herramientas de subversión cuenta con dos maneras de trabajar y radica en que se trabaje con exclusión mutua o son ella, es decir un desarrollador puede elegir un área del proyecto, bloquearla y trabajar sobre ella, al suceder esto solamente él puede trabajar sobre ella y también existe la posibilidad en que todos los desarrolladores trabajan de manera simultanea y la herramienta se encarga de unificar el trabajo de todos, validando posibles conflictos y resolviéndolos en los casos en que sea posible o solicitando la intervención de los desarrolladores cuando la herramienta no pueda decidir sobre las modificaciones.

# Evaluación de Herramientas de IC

## Aspectos a Evaluar

* Opensource.
* Soporte de lenguaje.
* Precio.
* Control de Bugs.
* Soporte para herramienta de control de versiones.
* Soporte para Publicaciones.
* Interfaz.
* Herramientas de Build que soporta.
* Herramientas de Pruebas que soporta.
* Dependencias y plataformas.

## Criterios Importantes

* Que sea una herramienta Opensource preferiblemente.
* Que tenga soporte para lenguaje Java.
* Que sea una herramienta gratuita.
* Que cuente con control y seguimiento de bugs.
* Que cuente al menos con soporte para Subversion, Git y Mercurial.
* Que cuente con un buen sistema de Publicación.
* Que cuente preferiblemente con una interfaz web.
* Que tenga soporte para Maven de preferencia.
* Que tenga soporte para pruebas unitarias, Junit de preferencia.
* Que tenga el mayor soporte de plataformas posible (Web de preferencia).

## Herramientas a evaluar

* Hudson.
* CruiseControl.

## Metodología de evaluación

Según nuestro criterio se cuenta con 10 aspectos importantes para los fines del desarrollo del proyecto, y serán evaluados como no cumple, bueno, muy bueno, excelente equivaliendo a 0, 4, 6, 10 respectivamente la evaluación será llevada a cabo respecto a cada aspecto de las herramientas y será justificada a través de la información encontrada sobre las mismas y cuando aplique sobre pruebas directas a la aplicación cuando sea posible, cada uno de los valores de las puntuaciones serán sumados y promediados para obtener una puntuación en un área de la herramienta y luego se mostrara de manera gráfica por cada herramienta la puntuación final, de la siguiente manera.

Al final de las evaluaciones presentaremos una gráfica comparativa final con la puntuación de cada una y basándonos en dicha gráfica y opiniones personales y convención nuestra elegiremos una herramienta para utilizar en el proyecto.

## Hudson

### Opensource

El objetivo principal de interesarnos en que la aplicación sea Opensource es por el constante crecimiento de las herramientas de este tipo, adicional a poder tener control total en aspectos como cambios o modificaciones deseadas por parte de nuestro equipo de trabajo en caso de ser necesario así como la posibilidad de hacer cambios en el código con el fin de arreglar posibles errores o mal funciones.

Según información encontrada en la web, Hudson es proyecto Opensource o de código abierto que tiene algunas partes como el diseño de la interfaz gráfica bajo licencia CreativeCommons y en lo que respecta a lo restante del proyecto bajo licencia MIT.

Adicionalmente en el tema de soporte la herramienta cuenta con una wiki muy completa de referencia y ayuda <http://wiki.hudson-ci.org/display/HUDSON/Home> que cuenta con topicos como sourcecode, issuetracker, developmentlifecycle, asi como información sobre plugins y otros temas, por lo que cuenta con buen soporte.

Por la información y licencia de la herramienta asignaremos un valor de excelente a esta área, que equivale a 10 punto

### SoportedelLenguaje

El lenguaje elegido por el equipo para desarrollo del proyecto y la aplicación es java, y el ide a utilizar es netbeans, que hace uso de acciones ant para la ejecución de sus proyectos y también cuenta con la posibilidad de crear proyectos basados en Maven, que es otra de las herramientas a utilizar por nosotros, estas son las restricciones que hemos considerado deberá cumplir como mínimo la herramienta para contar con posibilidad de garantizar una integración efectiva entre las diferentes tecnologías con las que hemos decidido trabajar.

Hudson cuenta con soporte para proyectos que están creados para funcionar con Apache Ant, Apache Maven, Comandos de consola, Groovy, Open MakeMeister, Apache Maven 2, Apache Maven 3, MsBuild, Nat y Rake.

Dando que se ha establecido utilizar proyectos en java generados para ser ejecutados con Maven y la gran cantidad de herramientas con las que es compatible Hudson hemos decidido dar un valor de excelente con 10 puntos en esta área.

### Precio

Lo ideal para el presupuesto y fines del proyecto es que la herramienta se distribuya de manera gratuita y Hudson cumple con esta condición, por lo tanto hemos elegido dar un punteo de excelente según nuestras expectativas y por lo tanto 10 puntos equivalentes a la nota en esta área.

### Control de bugs

Cuenta con compatibilidad de control de bugs con las herramientas Bugzilla, Jira, Rally y Trac, por la amplica cantidad de herramientas de control y rastreo de bugs y errores hemos decidido dar una nota de excelente a esta herramienta pues provee más de una opción y tiene soporte para herramientas de este tipo, por lo tanto tendremos una evaluación de 10 puntos en esta área.

### Soporte para Herramienta de Control de Versiones

Hudson cuenta con soporte para una gran cantidad de herramientas de control de versiones, principalmente las de interés para nosotros como equipo

* Subversion
* Mercurial
* Git

El resto de herramientas que soporta son:

* AccuRev
* AlienBrain
* Bazaar
* BitKeeper
* ClearCase
* CA Harvest
* CM Synergy
* CVS

Entre otras herramientas más, por el extenso soporte y por la compatibilidad con las opciones que hemos considerado damos una nota de excelente a esta área con 10 puntos.

### SoporteparaPublicación

Sobre soporte para publicación tiene compatibilidad con publicaciones via:

* Email
* Ejecutable
* FTP
* IRC
* Jabber
* Lotus Sametime
* RSS
* SCP
* Windows SystemTray

La opciones que consideramos adecuadas en relación a nuestro ambiente de trabajo son vía Email y Windows SystemTray, las opciones que ofrece Hudson son bastante funcionales y muchas más que otras herramientas, damos una puntuación de 10 a esta área.

### Interfaz

Cuenta con interfaz web, que es de fundamental importancia para nuestro criterio pues garantiza la portabilidad y dada la naturaleza de las preferencias de entorno de trabajo de nuestro equipo contamos con diferentes sistemas operativos para trabajar, entre la funcionalidad que ofrece la interfaz esta:

* Agregar proyectos y Jobs.
* Eliminar proyectos.
* Modificar proyectos.
* Detener Builds.
* Acceso a los artefactos de cada build.
* Gráfico de Historiales de builds.
* Pagina auto actualizable.
* Soporte para más de un proyecto.
* Vistas de más de un proyecto.
* Agregar ordenadores para builds distribuidos

En desventaja en este punto encontramos que no se puede pasar un build asignaremos muy bueno a esta sección con una puntuación de 6.

### Herramientas de Build que soporta.

Para fin del proyecto hemos definido utilizar Maven 3 porque no se necesita de ninguna otra herramienta para manejar las dependencias de los proyectos las herramientas con las que trabaja Hudson son las siguientes:

* Apache Ant,
* Apache Maven
* Comandos de consola
* Groovy
* Open MakeMeister
* Apache Maven 2
* Apache Maven 3
* MsBuild
* NAnt
* Rake.

En esta area asignaremos 10 puntos a la herramienta.

### Herramientas de Pruebas que soporta.

La herramienta de prueba planificada a utilizar y que interesan al equipo que sea soportada principalmente es JUnit para llevar a cabo pruebas unitarias de igual manera la mayor cantidad de herramientas a soportar es la mejor opción para el grupo, Maven tiene soporte para:

* Agitar
* CppUnit
* JUnit
* NUnit
* QualityCenter
* PHPUnit
* PMD
* Clover
* Selenium
* SilkCentral
* MSTest

Por cumplir con nuestro requerimiento principal y el extenso soporte asignamos un valor de 10 puntos.

### Dependencias y plataformas.

Hudson cuenta con compatibilidad con diferentes sistemas operativos, entre los que soporta encontramos Windows, Linux, también requiere de la máquina virtual de java JRE, corre sobre servidor Tomcat, en la mayoría de distribuciones de Linux no necesita de ninguna configuración adicional a la del paquete de instalación, en Windows requiere algunas configuraciones sobre las variables de entorno, pero son relativamente sencillas, trae incluidos de manera predeterminada plugins para manejar Apache ant, Maven 2, Maven 3 que son herramientas de build, la instalación y configuración es bastante sencilla. Por la relativamente media complejidad que implica en un ambiente Windows asignaremos un valor de 6 puntos, para una nota de muy bueno en esta area.

### Grafica de Resultados

El promedio obtenido por esta herramienta es 9.2

## CruiseControl

### Opensource

Es una herramienta distribuida bajo una licencia BSD y además es una herramienta gratuita, estos son los requerimientos mas importantes que buscamos como equipo en la herramienta en esta area, CruiseControl cumple con ambos, además cuenta con una wiki de ayuda <http://confluence.public.thoughtworks.org/display/CC/Home> donde se pueden encontrar una gran cantidad de tópicos y gran parte de la información a analizar en las próximas áreas de interés.

Dada esta información asignamos una calificación de 10 puntos a esta area.

### SoportedelLenguaje

CruiseControl esta implementado en Java, cuenta con una gran cantidad de plugins dado que el proyecto será desarrollado en base a proyectos java para ejecutarlos con Maven a travezde elidenetbeans la compatibilidad de CruiseControl con Maven es de fundamental importancia, las herramientas de build con las que tiene compatibilidad CruiseControl son Maven 2, Maven1, Ant, NAnt, make, scripts de terminal para Linux y Unix, y archivos .bat para Windows.

Dado que cuenta con soporte para Maven 2 y Maven 1 únicamente y no Maven 3 daremos una puntuación de muy buena a esta area, con 6 puntos.

### Precio

Es una herramienta gratuita cubriendo nuestro criterio de precio por lo tanto asignamos 10 puntos a esta área.

### Control de bugs

CruiseControl unicamente tiene compatibilidad con las herramientas de control y rastreo de erroes siguientes:

* Rally
* VersionOne

Si bien este no cuenta con muchas alternativas para el seguimiento de bugs si cuenta con soporte para el mismo, por lo tanto le daremos una puntuación de muy bueno con un valor de 6 puntos para esta área.

### Soporte para Herramientas de control de versiones

De las tres herramientas de mayor interés según nuestro criterio y necesidades CruiseControl unicamente tiene soporte para Subversion, una de las tres herramientas, todas las herramientas que soporta son las siguientes:

* Accurev
* Clearcase (dynamic views)
* Clearcase (snapshot views)
* Clearcase (UCM)
* CMSynergy
* jCVS
* MKS
* Perforce
* Pvcs
* StarTeam
* Subversion
* View CVS
* CVS (trigger a build via a tag)

Por lo tanto asignaremos un valor de Buena a esta área con una puntuación de 4.

### SoporteparaPublicaciones

Cuenta con una variedad de maneras de publicar sobre los resultados obtenidos, los mas importantes son los siguientes:

* Email
* FTP
* Jabber
* Lotus Sametime
* RSS
* SCP
* Windows System Tray
* Formatted Logging
* Yahoo Messenger
* X10

Dado que cuenta con una gran cantidad de opciones para notificar sobre los builds realizados definiremos una puntuacion de 10 a esta area.

### Interfaz

La interfaz de la aplicación es web punto importante para la portabilidad, ademas parte de la funcionalidad más importante que provee la interfaz de la aplicación es la siguiente:

* Permite Mostrar cambios y avances
* Permite agregar nuevos proyectos
* Permite elminar proyectos
* Permite modificar proyectos
* Permite denterbuilds en ejecución
* Permite acceso a los artefactos de los builds
* Historial grafico de builds
* Permite manejar varios proyectos
* Permite visualizar varios proyectos

Por el soporte amplio de operaciones de la interfaz gráfica asignamos una nota de 10 a esta área.

### Herramientas de Build que soporta.

Las herramientas de builds que soporta son las siguientes

* Maven 2
* Maven1
* Ant
* NAnt
* Make
* Scripts de terminal
* Archivos .bat

Soporta una gran cantidad de herramientas pero no Maven 3 a diferencia de la herramienta analizada con anterioridad por lo tanto asignaremos 6 puntos a esta área.

### Herramientas de Pruebas que soporta.

Las herramientas de prueba para las que tiene soporte CruiseControl son las siguientes:

* Agitar
* JUnit

Las pocas opciones de herramientas de pruebas limitan en gran manera el posible desarrollo de pruebas conforme se trabaja en un proyecto y a través del tiempo pues limitan únicamente al uso de las dos que tiene soporte cerrando la opción a nuevos tipos de pruebas, el soporte para herramientas de pruebas es bueno con 4 puntos en esta área.

### Dependencias y plataformas.

El uso de CruiseControl se facilita de gran manera al ser una interfaz web y optimiza su portabilida, cuenta con instalador de Windows, utiliza JRE pues está basado en java por lo tanto corre sobre servidor Apache Tomcat y es ejecutado por la JVM, utiliza mayormente Ant y Maven, por su portabilidad le hemos asignado 10 puntos a esta area.

### Grafica de Resultados

Promedio de 7.6

# Evaluación de Herramientas de control de Versiones

### Aspectos a Evaluar

* Facilidad de Uso
* Manejo de Branches
* Manejo de Merges
* Popularidad Soporte y Cross -plataform

### Criterios Importantes

* Que sea de la mayor facilidad posible de uso sin comprometer la calidad.
* Que tenga un manejo adecuado de Branches.
* Que realice Merges de la mejor manera posible sin ser demasiado complejo.
* Que exista buen soporte y una comunidad amplia en la web.

## Herramientas a evaluar

* Subversion
* Git

## Metodología de evaluación

Según nuestro criterio se cuenta con 10 aspectos importantes para los fines del desarrollo del proyecto, y serán evaluados como no cumple, bueno, muy bueno, excelente equivaliendo a 0, 4, 6, 10 respectivamente la evaluación será llevada a cabo respecto a cada aspecto de las herramientas y será justificada a través de la información encontrada sobre las mismas y cuando aplique sobre pruebas directas a la aplicación cuando sea posible, cada uno de los valores de las puntuaciones serán sumados y promediados para obtener una puntuación en un área de la herramienta y luego se mostrara de manera gráfica por cada herramienta la puntuación final, de la siguiente manera.

## Subversion

### Facilidad de Uso

Subversion se caracteriza por ser un sistema de control centralizado, es decir solamente existe una versión centralizada del proyecto y los usuarios deben actualizar sus cambios a esta versión o recurso centralizado, por lo tanto es más fácil para usuarios de subversion trabajar directamente en el trunk y actualizar la versión existente, esto implica una gran simplicidad, pues los cambios únicamente son agregados sin generarse ramas adicionales de manera innecesaria o cada vez que un desarrollador nuevo trabaje en el código. Asignaremos un valor de 10 a esta área por simplicidad y facilidad de uso.

### Manejo de Branches

Puede hacer checkin y chekcout parcial del repositorio, subversion no está orientado directamente al manejo de branches, no tiene soporte para tags, subversion se basa en almacenar las diferencias de los originales al crear un branch, al crear copias subversion o branches subversion utiliza un espacio muy pequeño en los cambios por ser estos únicamente almacenados como tal, como cambios y no como una versión nueva completa, permite branches y uniones entre los branches y uniones entre branches y trunk, por cumplir correctamente y ofrecer la funcionalidad necesaria a nuestro criterio evaluaremos con 10 esta área.

### Manejo de Merges

Los merges se pueden llevar a cabo entre branches y branches con el trunk esta es funcionalidad suficiente bajo nuestro criterio para evaluar con un 10 esta área.

**Popularidad Soporte y Cross -plataform**

Subversion es muy popular en la web y en herramientas de desarrollo, la mayoría de ides tienen soporte para subversion y tiene un buen nivel de Cross-plataform, pues existen varios clientes de subversion para varias versiones de sistemas operativos, esto aumenta su compatibilidad y popularidad, además cuentan con una wiki que es un buen soporte para la herramienta, asignaremos 10 puntos a esta área.

### Grafica

## Git

### Facilidad de Uso

La facilidad de uso no es el fuerte de Git, se dice que para utilizarlo hay implícita una curva de aprendizaje, es una herramienta de versionamiento descentralizada, es decir cada usuario cuenta con una versión completa del repositorio, la falta de simplicidad aunque puede ser una de las herramientas más avanzadas y funcionales esta fuera de nuestro criterio y por lo tanto asignaremos un valor de 6 puntos a esta área.

### Manejo de Branches

El manejo de branches en Git es bastante completo y sofisticado, casa usuario de la herramienta cuenta con una versión completa del repositorio en su ordenador por ser descentralizada, además los cambios a la aplicación se agregan generalmente luego de varios cambios y suelen ser cambios muy grandes para integrar, esta versión con la que cuenta cada usuario debe ser sicronizada frecuentemente con la vesion de una versión “Master” del repositorio para asegurarse de contar con las ultimas actualizciones. En el caso de Git debe ser copiada la totalidad del repositorio y no parcialmente como en subversion, por el manejo de branches y versiones por usuario de Git asignaremos 10 puntos a esta area.

### Manejo de Merges y commits

Los merges se llevan a cabo de manera similar a la de subversion pero hay que mantener una disciplina mayor pues el repositorio local requiere ser actualizado constantemente, la manera de identificar las versiones es un poco más compleja en Git que en subversion donde las versiones son numeraciones consecutivas, Git permite hacer commits con versiones intermedias no estables pues al final se hace un push para publicar la versión final funcional, por la robustez de Git para esta área 10 puntos.

**Popularidad Soporte y Cross -plataform**

Existen herramientas y versiones de Git para Mac, Windows y Linux, Git cuenta con una gran variedad de libros, tutoriales en línea de la página oficial así como videos, por estos motivos asignaremos 10 puntos a esta area.

### Grafica

# Conclusiones

## Resultados de Herramienta CI

### Hudson

**Promedio de 9.2**

### CruiseControl

**Promedio de 7.6**

La herramienta a utilizar para Integración Continua será Hudson, por sus ventajas en soporte de lenguaje control de bugs, control de versiones, soporte para herramientas de build y herramientas de pruebas que soporta.

## Resultados de Herramientas de control de Versiones

### Subversion

**Con unpromedio de 10**

### Git

**Con un promedio de 9**

La herramienta a utilizar según los valores de la evaluación basados en los criterios a cumplir establecidos por el equipo de trabajo será Sbuversion.

# Descripción de la herramienta seleccionada

El entorno de desarrollo estará compuesto de la siguiente manera

## Para desarrollo

* Kit de desarrollo Java (JDK)
* Framework Java Server Pages
* IDE para desarrollo en Java Netbeans

## Herramienta para manejo de Versiones

* Subversion como herramienta de control de versiones.
* Google code como servidor de subversion.

## Herramienta para Manejar Builds y Dependencias

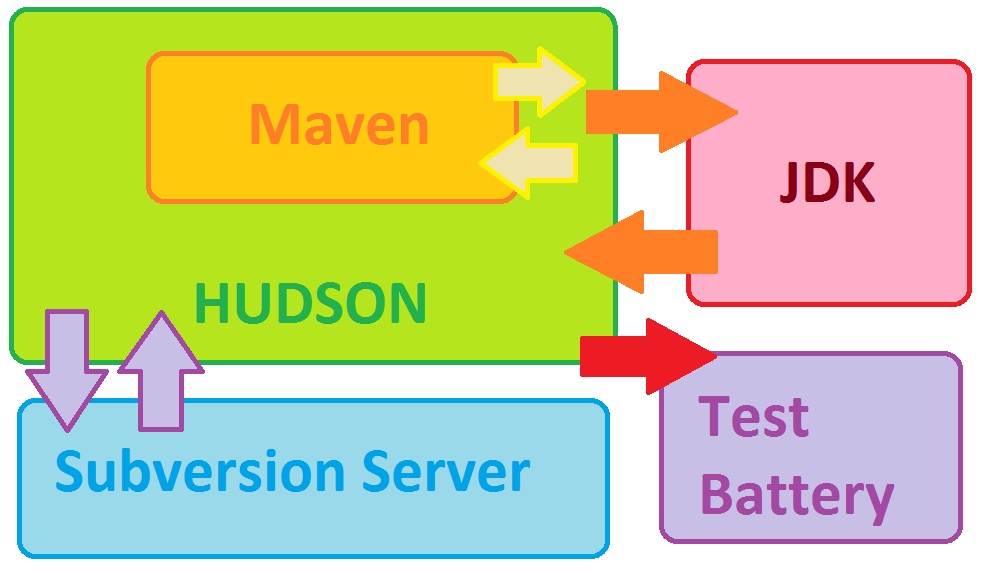
* Maven 3 como plugin de Hudson.
* Maven 3 como aplicación del Sistema Operativo.

## Herramienta para Manejo de Integracion continua

* Hudson para integrar las herramientas Maven, JDK y subversion

## Funcionamiento

Hudson es la herramienta que se encargara de controlar las conexiones entre jdk Maven y subversion server y se asegurara que se ejecuten las acciones de descargar del repositorio la versión que le sea indicada, gestionar dicha conexión, interactuar con Maven para interpretar el proyecto descargado y que se encargue de ejecutarlo compilarlo realizar las pruebas pre programadas y llevarlo al ambiente de despliegue, Maven vera este tipo de acciones como “Trabajos” los cuales son personalizables en varios aspectos como el tiempo máximo para eliminar los resultados de los builds, cuantos builds almacenar como máximo, que herramienta utilizar para ejecutar el build, entre otra gran variedad de opciones que pueden ser configuradas en el mismo.



# Estructura de directorios

## Definiciones, AcrónimosyAbreviaciones

* Directorio: Contenedor Virtual en la que se almacena un conjunto de archivos u otros directorios (Sub-directorios)
* Sistema de control de Versiones: Software que permite la gestión de cambios de los elementos de un producto de software (Código fuente, Configuración, Documentación etc.).
* Repositorio: Sitio centralizado donde de almacena la información.
* Pruebas Unitarias: Forma de probar que una porción de código funciona de manera correcta.
* Librerías(Bibliotecas): conjunto de sub programas que se utilizan para desarrollar software
* Código Fuente: conjunto de líneas de texto que representan las instrucciones que una computadora debe de seguir.
* Arquitectura de Software: diseño de más alto nivel de las estructuras de un sistema
* Base de Datos: Conjunto de Datos que se almacenan de forma persistente

## Estructura de directorios

Esta es la estructura de cómo se va a almacenar todo lo referente al proyecto en el sistema de control de versiones (subversion)

1. Libs: Este directorio contiene las librerías de terceros que utilizara organizadas por versiones
2. Tests: Este directorio contiene las diferentes pruebas Unitarias que se harán a lo largo del proyecto
3. Documentación: en este directorio se almacenara la documentación asociada al proyecto

* CU: es un sub-direcotriodonde se almacenara todo lo referente a los Casos de Uso del proyecto
* Arch: es un sub-directorio done se almacenará lo referente a la Arquitectura del proyecto

1. DB: Aquí se almacenara lo referente a la base de datos:

* SQLScript: sub Directorio doné se guarda el script de creación de la Base de Datos
* Querys: almacena los queries que se utilizaran en la aplicación

1. SRC: Este Directorio almacenara el código fuente del proyecto, cada sub Directorio representara un paquete individual

# Esquema de identificación de objetos

## Tipos de Archivos

### Archivos de configuración

### Estos archivos se refieren a los bach que sean necesarios crear para el funcionamiento correcto de la aplicación. Estos pueden ser creados para la instalación o para la creación de algún directorio necesario.

Este tipo de archivos serán de extensión .bash y su nombre ira antecedido por la aplicación ejemplo: seganaCreateDirectorio.bash

### Ejecutables:

Este tipo de archivos tendrán extensión .exe y en caso exista serán nombrado de la misma forma que los archivos “.bash” ejemplo: InstalerSegana.exe

### Clases

Las clases que se utilicen serán realizadas en java, la nomenclatura de las misma es descrita más adelante. Cabe destacar que para la comunicación con las jsp se utilizaran eterprise java been cuya nomenclatura es parecida a la de las clases.

### Paginas

Las páginas tendrán extensión jsp ya que estaremos trabajando con esta tecnología. De igual forma la nomenclatura para las jsp es descrita más adelante. Es importante mencionar que para la creación de estas páginas se utilizara lenguaje css y plantillas en html para hacer máságil la interfaz de la pagina.

### Base de datos

La base de datos que se tiene contemplado utilizar será mysql, en cuanto a la información de la nomenclatura que se estará utilizando esta viene descrita más adelante.

## Nombres

### Java:

#### Clases

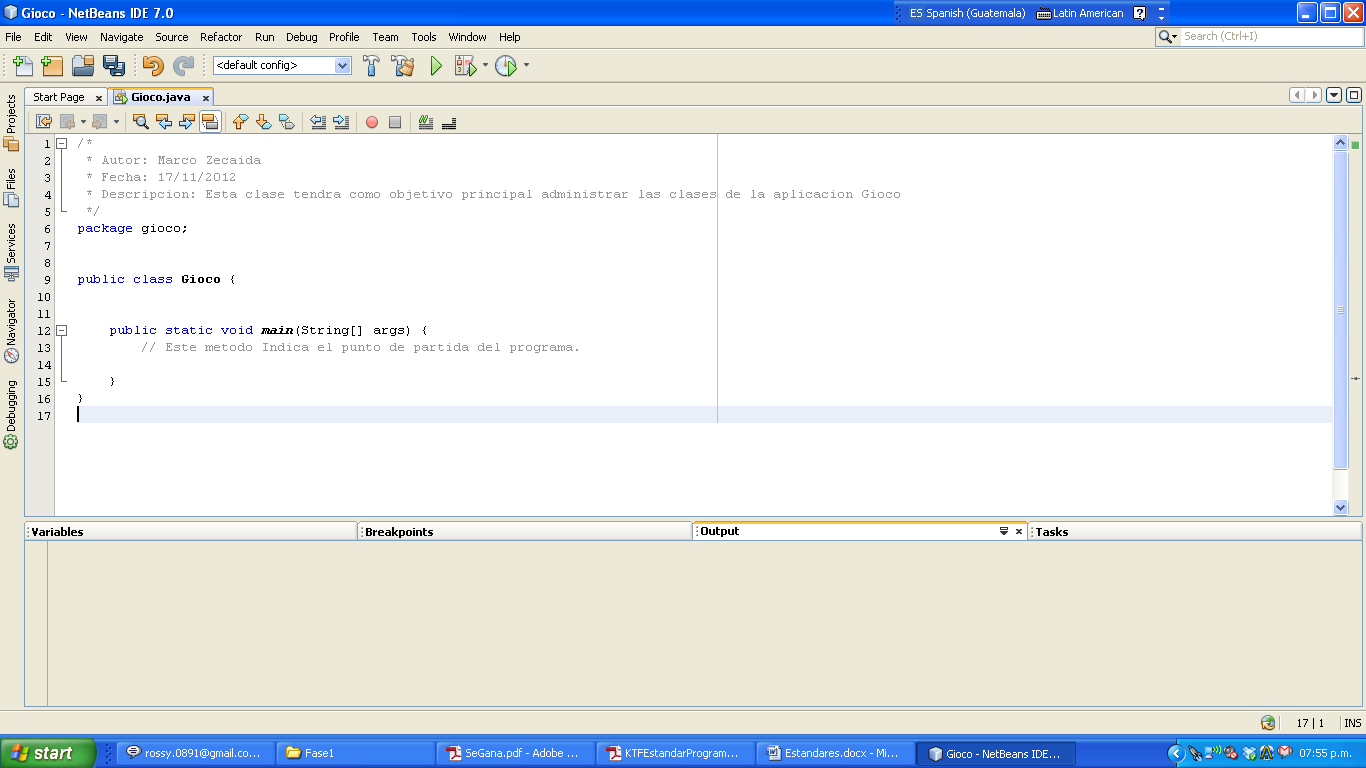
El nombre de las clases indicaran que es lo que hacen y para que sirven, dicho nombre debe de iniciar con mayúscula, el nombre del paquete indicara el modulo al que pertenece.

Se debe de colocar al inicio de cada clase la siguiente información:

Autor

* Fecha de creación
* Descripción breve de lo que hace.
* Así mismo se debe de colocar una descripción por cada método que se realice.

Ejemplo:



#### Nombre de variables, parámetros y constantes.

Los nombres de las variables deben de indicar el tipo de dato al que pertenecen y deben de ser acorde a lo que hacen. No se deben de declarar variables de esta forma:

Int variable1, variable2;

Así mismo se debe de inicializar las variables. La asignación quedaría de la siguiente forma:

* Variable int🡪intviContador = 0
  + V : variable, I: int, Contador: que hace la variable o para que sirve.
* Constante String 🡪StringcsSessionUsuario=”kibo”
  + C: constant, s: string, SessionUsuario: lo que hace y para que sirve.
* ParametroBoolean🡪booleanpbOpcionalidad
  + P: parametro, b: Boolean, Opcionalidad: Que es o para qué sirve.

#### Procedimientos y funciones

Se deben de colocar todos los procedimientos dentro de una clase como privados en la medida de lo posible.

Las funciones y los procedimientos se declaran de la siguiente forma:

* fiSuma() 🡪 f: de función, i: de int, Suma:descripción de la función.
* svCargaDatos() 🡪 s: procedimiento, v: void, Manejador: lo que hace el método.

### Jsp

Las jsp deben de llevar el nombre del modulo al que pertenecen+ “ \_ ” +<nombre de la pagina>

### Bases de datos

#### Tablas

Las tablas que serán declaradas en la base de datos deben de ser nombradas de la siguiente forma:

<modulo>+ “\_” + <nombre\_tabla>

#### Campos

El nombre de los campos estará precedido por el nombre de las tablas de manera abreviada.

Ejemplo:

Usr\_usuario, usr\_nombre, usr\_email🡪 son campos que pertenecen a la tabla usuario y que describen distinta información que será almacenada en la base de datos.

## Estándares

Se cuenta con los siguientes estándares:

### Estándares para la documentación:

#### Plantillas Rup

Se tiene contemplado que todos los archivos que tienen que ver con la documentación requerida para el cliente, estén estructurados en base a plantillas Rup. De esta forma facilitar la lectura de las mismas y permitirá un mejor control de versiones en los documentos.

#### Control de versiones

Se crearán nuevas versiones en caso de que sean agregadas funcionalidades que no han sido tomadas en cuenta en las anteriores versiones. Empezando por la versión 1.0 en adelante hasta y solo cambiara a versión 2.0 en caso de que se trate de una nueva versión general del proyecto.

### Metodología a utilizar

Para el desarrollo correcto de una aplicación es importante hace uso de una metodología que sirva de guía para la ejecución de las distintas tareas que comprende el sistema.

Durante la escogencia de la mejor metodología para este equipo de trabajo y para este sistema se tomaron en cuenta los siguientes puntos:

* La metodología debe de fortalecer las comunicaciones ya que los integrantes disponen de poco tiempo para reunirse.
* El proyecto debe de estar abierto a nuevos requerimientos, en este caso requerimientos que no hayan sido tomado en cuenta al inicio.
* Debe de dejar libertad a los desarrolladores y no se deben de establecer jerarquías ya que en este equipo de trabajo una de las filosofías es de que todos somos iguales.

En base a estos puntos se llegó a la conclusión que la mejor metodología que se podría utilizar es Scrum.

Ventajas Scrum

* Se plantean al inicio del sprint los objetivos que se desean alcanzar en el mismo, lo cual da lugar a una creación dinámica del diseño.
* Se organizaran reuniones diarias en las que habrá un moderador el cual realizara 3 preguntas básicas:
* ¿Qué hiciste ayer?
* ¿Qué harás hoy?
* ¿Hay algo que impida tu progreso?
* De esta manera se tendrá un mejor control de lo que hace cada desarrollador y ayuda al cumplimiento de las metas planteadas.
* Después de cada sprint habrá una reunión en la que se evaluara el desarrollo del proyecto lo cual nos determinara si algo se esta haciendo mal como corregirlo, y determinar qué cosas se están realizando de buena manera. Esto permitirá un desarrollo continuo.

# Esquema de nomenclatura de versiones

La nomenclatura estará dada por 3 números separada por puntos.

El Primer número indicara la versión actual.

Si la Versión es 0 indicara que está en versión Beta. Este número indicara cambios mayores a la proyecto conforme este valla evolucionando.

El segundo número es el número de cambios menores que se hacen a la versión actual. Este número estará desde 1 hasta 20

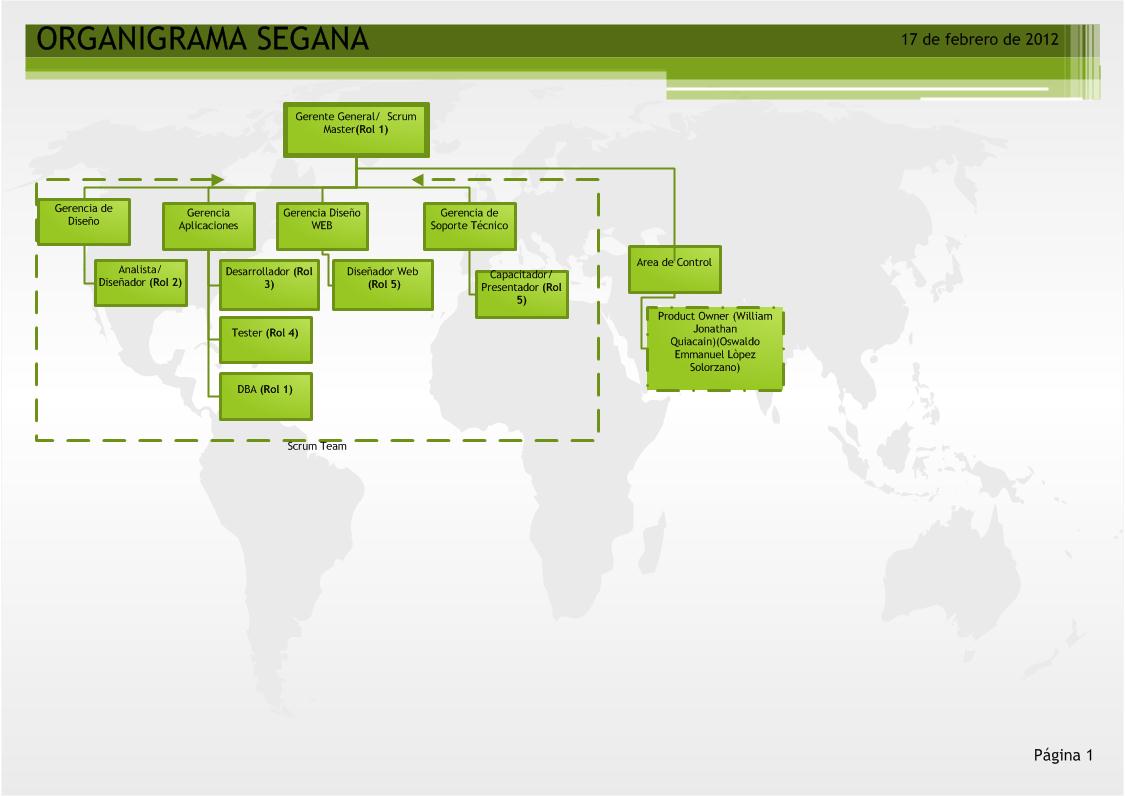
El tercer número será para indicar mantenimiento este número variara desde 1 a 20.

**Cambio Mayor.[[Cambio Menor].Mantenimiento]**

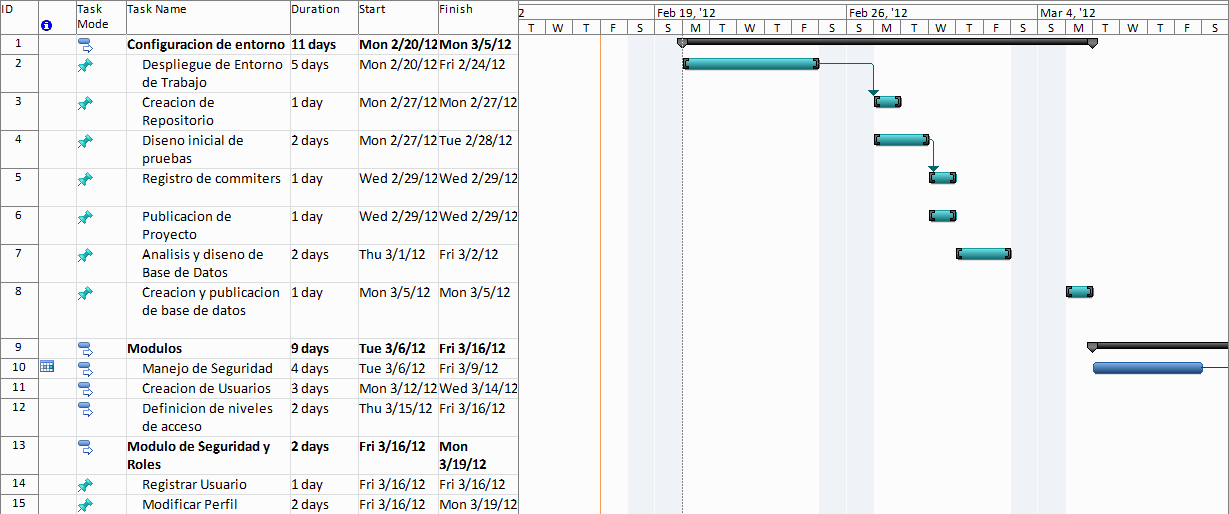
# Proceso de liberación

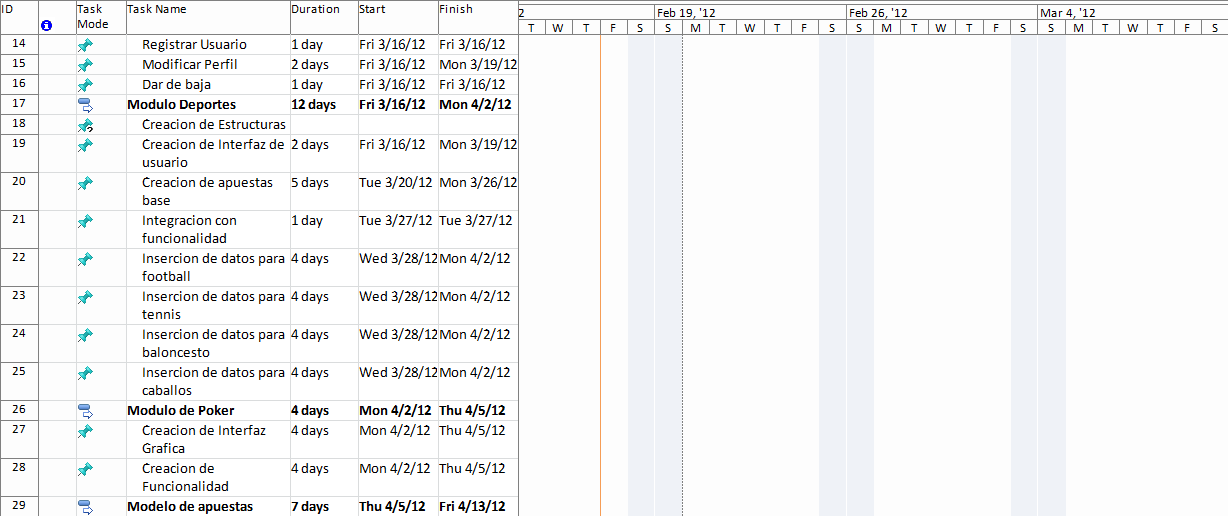
## Responsabilidades

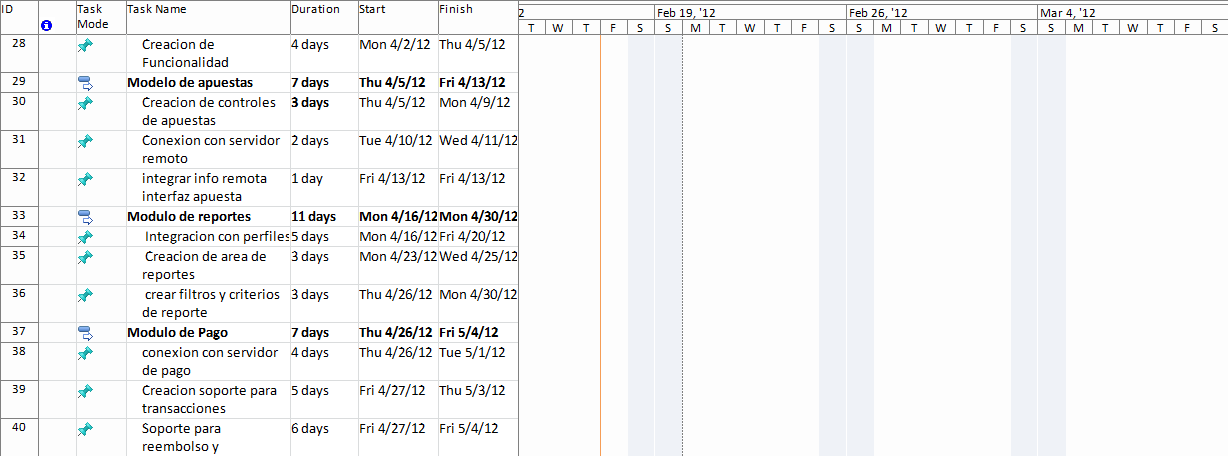
### Organigrama



## Programa (calendario, horario)







## Plan de costos

### **Hardware**

- Servidor de Integración Continua (ya adquirido) 0.00

- Servidor de Producción (ya adquirido) 0.00

- 5 estaciones de trabajo (ya adquiridas) 0.00

### **Software**

### (Se utilizaran herramientas open source):

- Software de Integración Continua (Hudson) 0.00

- IDE (Netbeans) 0.00

- Pruebas Unitarias 0.00

- Constructor de Aplicaciones (Maven) 0.00

- JUnit (biblioteca para pruebas unitarias) 0.00

### **Servicios**

(factor de 10% de aumento mensual):

- Internet 400.00 mensuales

- Energía Eléctrica 200.00 mensuales

- Agua Potable 150.00 mensuales

- Servicio de Limpieza 50.00 mensuales

### **Personal**

|  |  |
| --- | --- |
| Desarrolladores | Pago x Hora |
| Scrum Master | 37.00 |
| Analista Diseñador | 35.00 |
| Desarrollador | 35.00 |
| Tester | 35.00 |
| DBA | 35.00 |
| Diseñador Web | 35,00 |
| Capacitador | 35.00 |

### **Capacitación**

- Administrador 400.00 por semana

- Corredor 100.00 por semana

- Moderador 100.00 por semana

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Semana 1 | Semana 2 | Semana 3 | Semana 4 | Semana 5 | Semana 6 | Semana 7 |  |
| Agua Potable | 37.5 | 37.5 | 37.5 | 37.5 | 41.25 | 41.25 | 41.25 | 273.75 |
| Consumo de Energía | 50 | 50 | 50 | 50 | 55 | 55 | 55 | 365 |
| Servicio de Internet | 100 | 100 | 100 | 100 | 110 | 110 | 110 | 730 |
| Servicio de Limpieza | 12.5 | 12.5 | 12.5 | 12.5 | 13.75 | 13.75 | 13.75 | 91.25 |
| Scrum Master | 1480 | 1480 | 1480 | 1480 | 1480 | 1480 | 1480 | 10360 |
| Analista Diseñador | 1440 | 1440 | 1440 | 1440 | 1440 | 1440 | 1440 | 10080 |
| Desarrollador | 1400 | 1400 | 1400 | 1400 | 1400 | 1400 | 1400 | 9800 |
| Tester | 1280 | 1280 | 1280 | 1280 | 1280 | 1280 | 1280 | 8960 |
| DBA | 1440 | 1440 | 1440 | 1440 | 1440 | 1440 | 1440 | 10080 |
| Diseñador Web | 1400 | 1400 | 1400 | 1400 | 1400 | 1400 | 1400 | 9800 |
| Capacitador | 1360 | 1360 | 1360 | 1360 | 1360 | 1360 | 1360 | 9520 |
| Capacitación Administrador |  |  |  | 400 | 400 | 400 | 400 | 1600 |
| Capacitación Corredor |  |  |  |  |  | 100 | 100 | 200 |
| Capacitación Moderador |  |  |  |  |  | 100 | 100 | 200 |
| Total Por semana | 10000 | 10000 | 10000 | 10400 | 10420 | 10620 | 10620 | 72060 |

### **Costo aproximado**

Un Costo Aproximado del Proyecto de Q 72060.00

## Capacitación

Se tiene contemplado realizar un proceso de capacitación tanto a nivel de usuarios finales como también al personal encargado de dar mantenimiento a la aplicación, en nuestro caso son: Administrador, Corredor, Moderador.

### **Metodología de Capacitación**

La capacitación será una actividad sistemática, planificada y permanente que se llevaría a cabo con el objetivo de preparar, desarrollar e integrar los recursos humanos con los que se cuenta. Mediante la entrega de conocimientos, desarrollo de habilidades y actitudes necesarias para el mejor desempeño de todos los trabajadores en sus actuales y futuros cargos y adaptarlos a las exigencias cambiantes del entorno. Algunas preguntas que deben ser formuladas son:

1. Que es lo que lo que los trabajadores hacen que no deben hacer
2. Que deberían hacer y como
3. Que deben estar en capacidad de hacer los trabajadores cuando finalice

Su capacitación.

1. Que conocimientos, o manejo de instrumentos considera que deben incluirse en el contenido de la capacitación
2. como se van a incorporar los aspectos de capacitación al trabajo
3. como se va a dar seguimiento una vez finalizada la capacitación.

**Pasos a llevarse a cabo:**

Para la realización de la capacitación del personal adquirido o con el que se cuenta, se dividirían en:

Inducción: Cuando contamos con un nuevo elemento, es necesario orientarlo y capacitarlo, proporcionándole la información y los conocimientos que son necesarios para que tenga éxito en su nueva posición, aun cuando ya se encuentre laborando en el puesto y tenga experiencia.

Entrenamiento en el puesto: una vez finalizado el paso de inducción el siguiente paso es el entrenamiento en el puesto, el nuevo elemento requiere entrenamiento especifico sobre el puesto que va a desempeñar.

**Entrenamiento en cada Puesto:**

**Administrador:** La persona encargada de fungir el puesto de Administrador será capacitada para poder manejar entre otras funciones: las actividades de los clientes y corredores, respuestas a quejas de los clientes y corredores. Se dara cierto tipo de adiestramiento en el funcionamiento de la aplicación para que pueda familiarizarse con cada una de las funciones, se pondrá en un periodo de prueba donde se someterá a escenarios reales, y poder llevar a cabo diferentes funciones.

Se pondrá a su disposición todo tipo de documentación, manuales, documentos audiovisuales etc..

La función del administrador es una función muy importante por lo que una vez finalizada la capacitación es importante dar un seguimiento del desenvolvimiento del mismo.

Tiempo Estimado: 20 dias

Horas/dia: 4

Costo Estimado: 20.00 Q por Hora

**Corredor:** La persona que cumpla con esta función tiene entre sus labores el establecer las cuotas y cerrar apuestas, es un papel relativamente sencillo pero de alta importancia, dentro de la capacitación esta, poner a disposición del trabajar documentación que describa el funcionamiento de la aplicación para poder llevar a cabo sus funciones, también documentos audiovisuales, y asesoramiento que guie en el proceso de capacitación, someter al trabajador a escenarios reales.

Tiempo Estimado: 10 dias

Horas/dia: 2

Costo Estimado: 10.00 Q por Hora

**Moderador:** la persona que tenga a su cargo la función de moderador recibirá dentro del plan de capacitación un documento que describa cada una de las funcionalidades de este rol y material audiovisual que ejemplifique el uso de estas funciones, asi mismo se contara con capacitación presencial, y se someterá a escenarios reales.

Tiempo Estimado: 10 dias

Horas/dia: 2

Costo Estimado: 10.00 Q por Hora

**Cliente:** los clientes son los usuarios finales que hará unos de la aplicación, estos usuarios tendrán una capacitación especial, se pondrá a disposición de los usuarios dentro de la página principal de la aplicación una sección de ayuda, donde se podrán hacer descargas de documentos que incluyen manuales y video tutoriales acerca del funcionamiento de la aplicación. Se pondrá a disposición de estos usuarios una sección de dudas donde se podrán expresar cualquier inquietud que se tenga acerca del funcionamiento.

# Pruebas de concepto

<escriba aquí…>