TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica

Escuela de Computación
Curso: IC6831
Aseguramiento de la Calidad del Software
Grupo 2, II Semestre 2017
Profesor: Saúl Calderón Ramírez

Tarea 3:
Administración de la Configuración del Software

Integrantes:
Brandon Dinarte 2015088894
Armando López 2015125414
Andrey Mendoza 2015082908
Julian Salinas 2015114132

Domingo 15 de octubre del 2017
ITCR, Sede Central Cartago

1. Introducción

El presente documento define un plan de administración de la configuración del software a ser aplicado durante la realización del proyecto semestral del curso de Aseguramiento de la Calidad del Software. Incluye diversos apartados relacionados a los ítems de configuración del software, como los aspectos de definición de los mismos, sus procedimientos de revisión e implementación de cambios, entre otros. Parte de la asignación se encuentra implementada en un repositorio de una plataforma basada en Git, cuyo enlace no se adjunta con el presente documento.

2. Ítems de configuración del software

Un proyecto de software está compuesto por diferentes ítems de configuración que son utilizados para construir una versión específica exacta del sistema en el momento que se desea. A continuación, se muestra una tabla con los ítems de configuración utilizados en el proyecto de Autocaras y las siguientes características para cada uno:

- **Tipo:** se refiere a la etapa a la que pertenece el ítem. Las categorías son:
  - *Análisis de requerimientos:* documentos de requerimientos, atributos de calidad, enunciados, entrevistas.
  - *Diseño:* diagrama de componentes, diagrama UML.
  - *Implementación:* código fuente, herramientas como starUML, eclipse, junit, etc.
  - *Pruebas*
- **Detalle:** clarifica a qué elemento especifico del tipo de item se refiere. Por ejemplo, un item de tipo diseño puede tener el detalle del diagrama de componentes o diagrama UML.
- **Nombre:** será el nombre del recurso, tal cual estará disponible en el repositorio.
- **Nombre y versión de herramienta editora:** detalles del programa que permite realizar cambios en el item.
- **Volatilidad:** se refiere a la tasa de cambios por sprint del ítem específico. Se clasifica en 3 categorías:
  1. **Baja:** no hay cambios.
  2. **Media:** se utiliza durante más de un sprint.
  3. **Alta:** sufre cambios cada sprint.

Cuadro 1: Ítems de Configuración del Software

Número	Tipo	Detalle	Nombre	Herramienta editora	Volatilidad
1	Análisis de Requerimientos	Documento de requerimientos	Especificación de Requerimientos del sistema	Overleaf Real Time Writing and Publishing Tool 2017	Baja
2	Análisis de Requerimientos	Atributos de calidad	Requerimientos de Calidad	Overleaf Real Time Writing and Publishing Tool 2017	Baja
3	Análisis de Requerimientos	Atributos de calidad	Estándar de Codificación	Overleaf Real Time Writing and Publishing Tool 2017	Baja
4	Diseño	Diagrama de clases	Diagrama de Clases	Draw.io Flowchart Maker & Online Diagram Software	Media

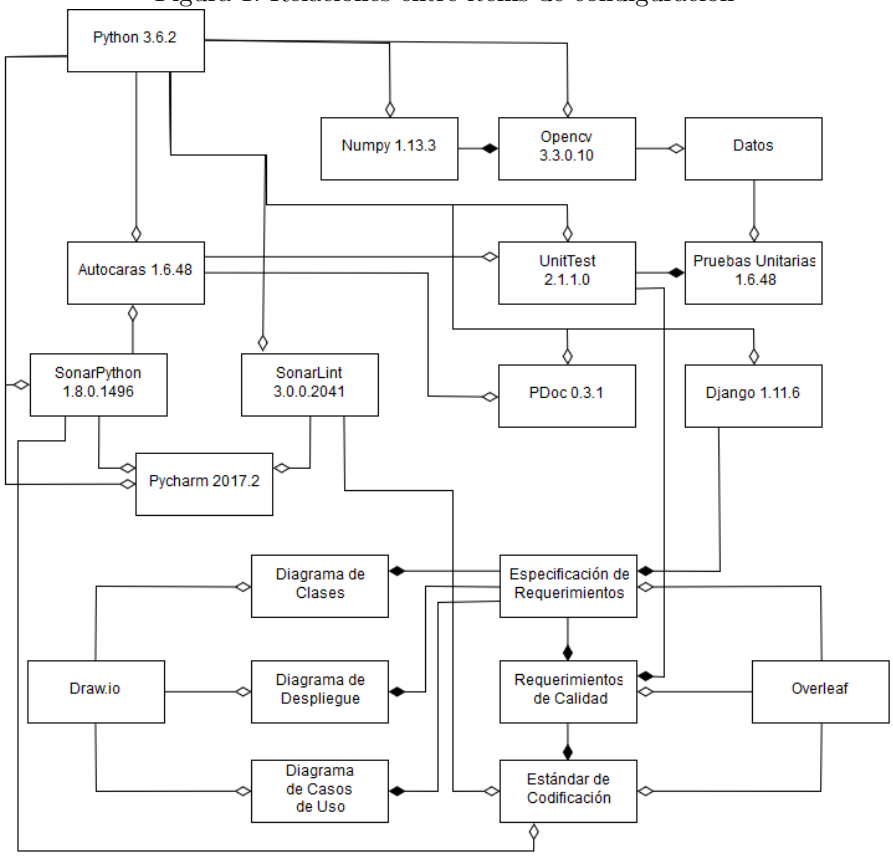
5	Diseño	Diagrama de despliegue	Diagrama de Despliegue	Draw.io Flowchart Maker & Online Diagram Software	Baja
6	Diseño	Diagrama de casos de uso	Diagrama de Casos de Uso	Draw.io Flowchart Maker & Online Diagram Software	Baja
7	Implementación	Pruebas	UnitTest 2.1.1.0	PyCharm: Python IDE for Professional Developers, v2017.2	Baja
8	Implementación	Editor de código	Pycharm 2017.2 (URL)	Sin herramienta	Media
9	Implementación	Biblioteca	SonarLint 3.0.0.2041 (URL)	PyCharm: Python IDE for Professional Developers, v2017.2	Baja
10	Implementación	Biblioteca	SonarPython 1.8.0.1496 (URL)	PyCharm: Python IDE for Professional Developers, v2017.2	Baja
11	Implementación	Biblioteca	Opencv 3.3.0.10 (URL)	PyCharm: Python IDE for Professional Developers, v2017.2	Baja
12	Implementación	Biblioteca	Numpy 1.13.3 (URL)	PyCharm: Python IDE for Professional Developers, v2017.2	Baja
13	Implementación	Biblioteca	Django 1.11.6 (URL)	PyCharm: Python IDE for Professional Developers, v2017.2	Baja
14	Implementación	Intérprete	Python 3.6.2 (URL)	PyCharm: Python IDE for Professional Developers, v2017.2	Baja
15	Implementación	Biblioteca	PDoc 0.3.1 (URL)	PyCharm: Python IDE for Professional Developers, v2017.2	Baja
16	Implementación	Código Fuente	Autocaras 1.6.48 (URL)	PyCharm: Python IDE for Professional Developers, v2017.2	Baja
17	Implementación	Editor de LaTeX	Overleaf (URL)	Sin herramienta	Baja
18	Implementación	Editor de diagramas	Draw.io (URL)	Sin herramienta	Baja

19	Pruebas	Pruebas unitarias	Pruebas Unitarias 1.6.48	PyCharm: Python IDE for Professional Developers, v2017.2	Baja
20	Pruebas	Datos	Datos (URL)	Sin herramienta	Baja

2.1. Relación entre ítems de configuración definidos

A continuación, se muestra un diagrama básico de composiciones y dependencias con el objetivo de visualizar las relaciones entre los ítems de configuración definidos para el proyecto *Autocaras*.

Figura 1: Relaciones entre ítems de condiguración



2.2. Versionamiento de las lineas base

La mayoría de los proyectos tiene dependencias externas en las que apoyan su desarrollo, incluso es posible modular los proyectos para que su mantenimiento pueda ser independiente o utilizable por otros proyectos. Por esta razón, la información de las actualizaciones o cambios sobre estas fuentes debe ser lo mas precisa y consistente posible.

Para lograr lo anterior, se hace uso del estándar de versionamiento semántico propuesto por Tom Preston Werner, cofundador de Github. La información respecto a este estándar puede ser revisada en el siguiente enlace `http://semver.org/`.

El sistema debe tener un número de versión con el formato X.Y.Z, donde X,Y,Z son número enteros positivos. X es un contador usado para identificar la versión del *API*. Cuando ocurren cambios que no son compatibles con la versión anterior, se debe incrementar este contador. Si el proyecto se encuentra en una etapa de desarrollo inicial se puede definir X igual a cero. Por otra parte Y hace referencia a la cantidad de funcionalidades que han sido implementadas. Si se incorpora una nueva funcionalidad que es retrocompatible con la versión anterior del sistema, el contador Y es incrementado. El último elemento Z es usado para identificar la cantidad de errores pequeños (que no afectan la compatibilidad con la versión anterior) que han sido solucionados o cambios que

optimizan alguna funcionalidad pero que no alteran su modo de operar.

2.2.1. Versión actual del sistema

La versión actual del sistema de reconocimiento facial corresponde a la *1.6.49* según el formato X.Y.Z mencionado anteriormente.

El contador inicial (X) comenzó desde 0, pues como se menciona en *semver.org*, dicho contador en 0 significa que cualquier cosa puede cambiar en cualquier momento y el API pública no debiera ser considerada como estable. Una vez que las tecnologías fueron probadas se llegó a un consenso con relación a la estructura, diseño y tecnologías a utilizar, permitiendo de esta forma aumentar dicho contador a 1.

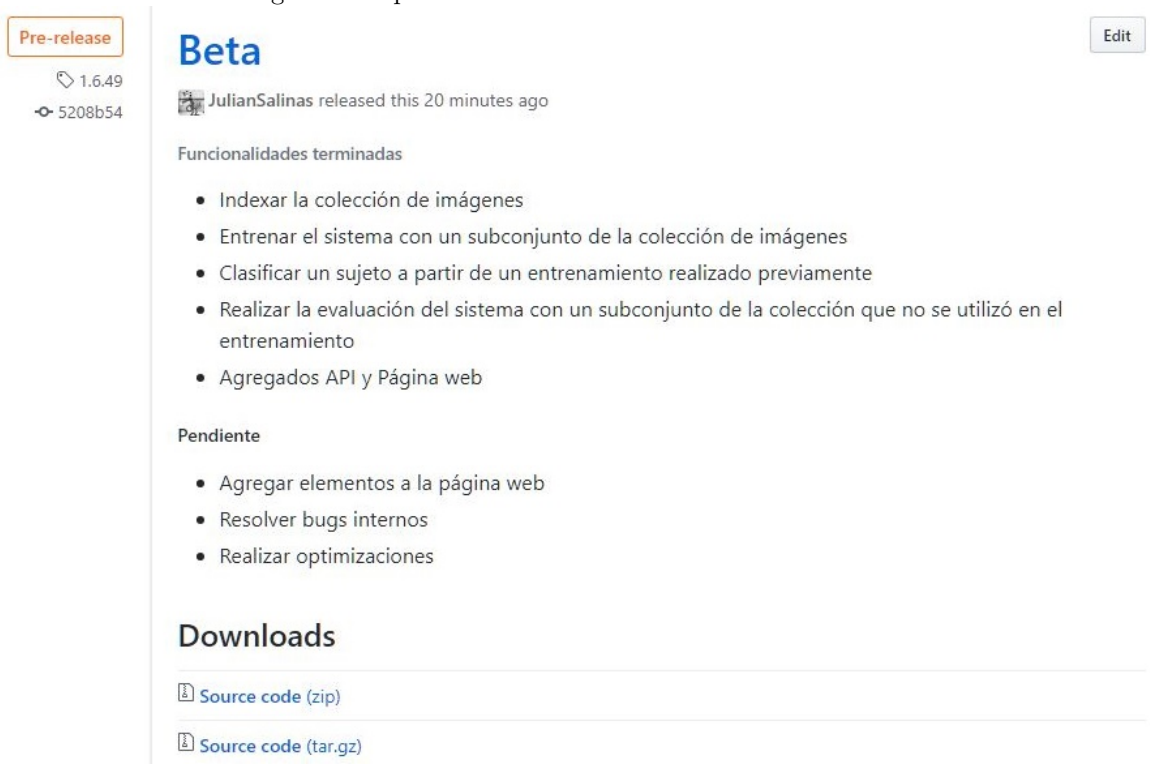
El segundo contador (Y) se basó en la introducción de nuevas funcionalidades. Para incrementar este contador se tomaron en cuenta la carga e indexación de imágenes presentes en la base de datos, el entrenamiento del sistema, la clasificación de sujetos, la creación de informes acerca de la evaluación del sistema con una cantidad reservada de imágenes de la base de datos, una página web y un API que permite la comunicación de la página con el modelo.

El último contador (z) corresponde a la cantidad de optimizaciones realizadas y de errores corregidos. También se hizo un incremento del contador cuando partes del sistema omitidas temporalmente fueron implementadas, por ejemplo, validar que una cadena corresponde a un directorio o archivo del sistema, manejo de errores al escribir archivos, actualizaciones importantes del *readme*, entre otros.

El sistema se encuentra fase Beta, pues representa la primera versión completa del mismo, es decir, todas las funcionalidades prioritarias han sido implementadas, sin embargo, es posible que sea inestable pero útil para que sea considerada como una versión preliminar suficiente para realizar pruebas con los involucrados.

El propósito de la versión Beta para este caso en específico es involucrar más a los usuarios finales para que realicen pruebas, permitiendo encontrar errores que desde el punto de vista de los desarrolladores no son notables y obteniendo consejos para el mejoramiento del sistema, así como, sugerencias para futuras funcionalidades.

Figura 2: Captura de la versión del sistema desde Github



2.3. Tutorial de uso para la herramienta de gestión de repositorios

Se hace uso de la herramienta para la gestión de repositorios *Gitkraken 3.1.1*, un cliente de Git construido en Electron. Esta puede ser obtenida desde su página oficial `https://www.gitkraken.com/download`. Esta herramienta permite clonar o inicializar repositorios ya sean locales o remotos, resolver conflictos al momento de fusionar ramas, etiquetar versiones del sistema, visualizar cambios en los archivos de la configuración y de forma general, permite interactuar de una forma más eficiente con los procesos más comunes que se llevan a cabo con Git.

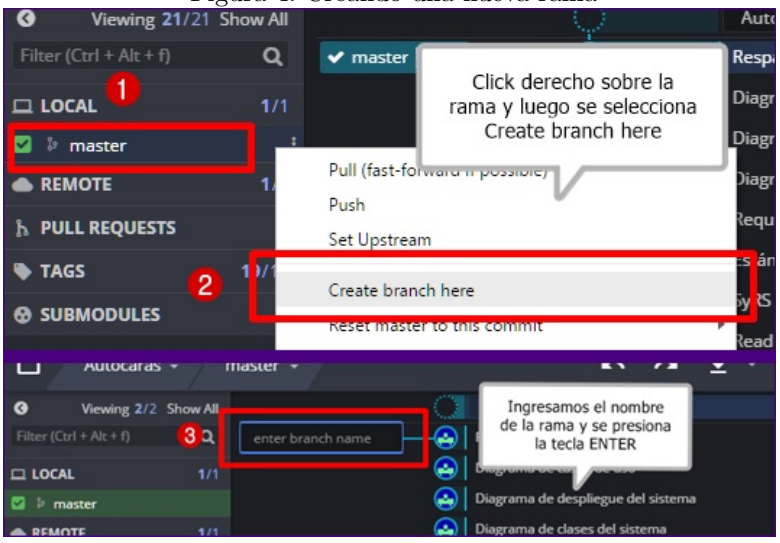
2.3.1. Tutorial para la creación de una rama local

Las ramas en Git son como ramas en los árboles, tienen que ramificarse de otra. Cada nuevo proyecto comienza con una rama predeterminada llamada *master*, y es de esta de donde se podrá crear una nueva rama. Una vez descargada e instalada la herramienta es necesario clonar el repositorio. Para ello se debe abrir *Gitkraken* y presionar la combinación *Ctrl+N*, esto nos mostrará una ventana donde deberemos ingresar la url del repositorio remoto y la ruta donde se guardará localmente. Una vez ingresados los datos únicamente presionamos el botón *Clone the repo!*. Después de esto el programa nos mostrará un mensaje informando el éxito o fallo de la operación. Ver figura 3.

Figura 3: Clonando repositorio

El primer paso para crear una rama usando Gitkraken es colocar el cursor sobre la rama *master* y abrir el menú contextual con *click* derecho, luego se presiona la opción *Create branch here*. Esto nos dirigirá a una caja de texto donde se debe escribir el nombre de la nueva rama. Al escribir el nombre se presiona la tecla *Enter*, justo después de hacerlo el programa mostrará un mensaje informando el éxito o fallo de la operación. Ver figura 4.

Figura 4: Creando una nueva rama



Al crear una nueva rama usando Gitkraken, el espacio de trabajo cambiará hacia la nueva rama. Es decir, crear una nueva rama es equivalente a ejecutar el comando *git checkout -b NombreRamaNueva* desde la linea de comandos de Git.

2.3.2. Actividades de aseguramiento previas a creación de una línea base

Cada categoría de ítem de configuración de software requiere un conjunto de actividades diferentes para asegurar su calidad antes de incorporar los cambios realizados a la línea base del proyecto. Se presentan dichas

actividades según el tipo de ítem que implementa los cambios.

Ítem de análisis de requerimientos

Para integrar los cambios y avances realizados a la línea base del proyecto, cuando se trata de ítems de análisis de requerimientos es necesario completar las siguientes actividades.

1. Validación de cambios de requerimientos con el usuario. Debe establecerse una reunión ya sea personal o no, con el cliente para que el mismo dé su aprobación sobre los cambios realizados a los requerimientos.
    - Documentos de referencia. Algunos procedimientos de cambio de requerimientos se establecen en el documentos de *Requerimientos de calidad*, así como en el documento de *Especificación de requerimientos del sistema*. Siendo el segundo un documento más enfocado a los requerimientos funcionales definidos por el usuario.
2. Documentación de los aspectos relacionados al cambio del ítem. Debe completar un proceso de respaldo de la información relacionada al cambio del requerimiento, lo cual incluye aspectos como la plantilla de información de cambio.
    - Documentos de referencia. El documento de *Requerimientos de la calidad*, define una plantilla sencilla para la documentación de cambios en los requerimientos. De igual forma en el proceso se ve incluido el documento de *Especificación de requerimientos del sistema*.

Ítem de diseño

En términos de ítems de diseño, las actividades que se deben completar para integrar los cambios y avances realizados a la línea base del proyecto incluyen:

1. Análisis de viabilidad por parte del equipo de desarrollo. El equipo debe ser capaz de implementar los cambios definidos en el diseño, teniendo en cuenta los recursos y capacidades disponibles en el período afectado por dicho cambio.
    - Documentos de referencia. Los documentos que proporcionan información acerca de los procedimientos de cambio para este tipo de ítem son el de *Requerimientos de calidad*, el documento del *Estándar de codificación* y también se ven afectados los documentos de *Diagramas de casos de uso, de clases, de despliegue*.
2. Validación del diseño contra la especificación de requerimientos. Los cambios realizados en el diseño deben pasar por un análisis para su validación por parte del arquitecto encargado, todo cambio debe estar contemplado junto con sus consecuencias en los requerimientos en caso de existir.
    - Documentos de referencia. El principal documento que proporciona información acerca de los procedimientos de cambio para este tipo de ítem es el de *Requerimientos de calidad*. De igual forma estará implicado el *Diagrama de clases del sistema y Diagrama de despliegue*. El documento de *Especificación de requerimientos del sistema* también juega papel importante al ser el marco de la comparación para la validación de los cambios.

Ítem de implementación

Para integrar los cambios y avances realizados a la línea base del proyecto, cuando se trata de ítems de implementación es necesario completar las siguientes actividades.

1. Cumplimiento del estándar de codificación. Toda implementación de cambios debe estar sujeta bajo los puntos definidos en el estándar de codificación.
    - Documentos de referencia. El documento del *Estándar de codificación*.
    - Herramientas utilizadas. Se utiliza la herramienta de SonarQube para el análisis estático del código antes de su aceptación.
2. Ejecución exitosa de todas las pruebas unitarias, lo cual se interpreta como la correcta codificación de los cambios de implementación realizados.
    - Documentos de referencia. Los documentos que proporcionan información acerca de los procedimientos de cambio para este tipo de ítem son el de *Requerimientos de calidad* y el documento del *Estándar de codificación*.

- Herramientas utilizadas. Ente las herramientas que se utilizan para la ejecución de pruebas se encuentra la biblioteca de Python *Pytest*, que sirve de marco para las pruebas unitarias.

3. Validación de la codificación conforme al diseño. Consiste en el análisis y comparación entre el ítem de implementación y su ítem de diseño correspondiente.

    - Documentos de referencia. Según sea el ítem de implementación su comparación debe ser contra uno de los documentos mencionados, siendo estos *Diagrama de clases del sistema*, *Diagrama de casos de uso*, *Diagrama de despliegue del sistema*.

Ítem de pruebas

En términos de ítems de pruebas, las actividades que se deben completar para integrar los cambios y avances realizados a la línea base del proyecto incluyen:

1. Seguimiento del procedimiento de cambio para pruebas. Las pruebas que cumplan con lo especificado para su proceso de alteración pueden formar parte de la línea base del proyecto.

    - Documentos de referencia. El documento de *Requerimientos de calidad* especifica el procedimiento de cambio y la plantilla de documentación para el mismo.

2. Compilación y resultados exitosos. Las pruebas serán analizadas para verificar que cumplan con su propósito de forma correcta antes de ser aceptadas.

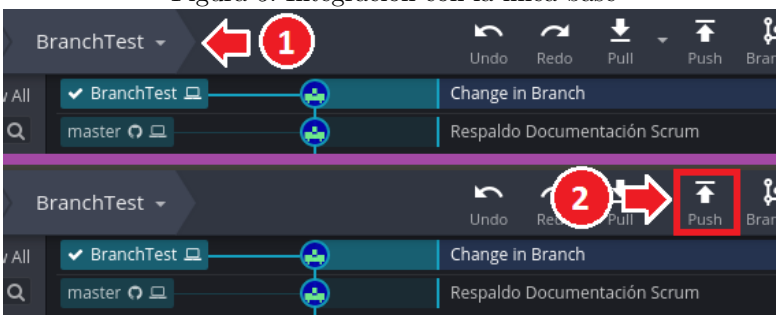
    - Herramientas utilizadas. Ente las herramientas que se utilizan para la ejecución de pruebas se encuentra la biblioteca de Python *Pytest*, que sirve de marco para las pruebas unitarias.

2.3.3. Creación y reconstrucción de una línea base

Creación

Una vez se han completado las actividades de aseguramiento de la calidad para la creación de líneas base de forma exitosa y dando por sentado que se trabaja sobre una rama a nivel local, el procedimiento de aplicación de los cambios a la línea base para su creación requiere primero de asegurarse que el espacio de trabajo sea la rama deseada, siendo en el ejemplo *BranchTest*, para lo cual se selecciona en la pestaña señalada por el paso #1 en la figura 5. Luego para integrar los cambios debe seleccionarse la opción de *Push*, señalada en el paso #2 de la figura 5. Con esto pueden recibirse diferentes respuestas, ya sea de éxito, de conflictos en la integración u errores propios de permisos o conexión. Posteriormente puede etiquetarse la línea base con el procedimiento establecido en el tutorial para el etiquetado de versiones.

Figura 5: Integración con la línea base



Reconstrucción

Para obtener una versión de línea base del proyecto, existen dos opciones. La primer opción es sencilla pero limitada y consiste en simplemente clonar el proyecto, con lo cual se obtiene la última versión disponible del mismo, para este procedimiento puede visitarse el tutorial para creación de una rama local que lo demuestra de forma general. La segunda opción es posicionarse es alguna de las etiquetas de versión definidas, y abrir el menú contextual con el botón derecho del mouse como se aprecia en el punto #1 de la figura 6. Luego seleccionar la opción de *Reset BranchName to this commit* en el paso #2, que reestablece la rama actual a la versión seleccionada anteriormente con la etiqueta, esta opción a su vez genera tres opcionas más específicas para proceder. De las tres opciones disponibles para el reestablecimiento interesa la de *Hard - Discard all changes* para deshacerse de todo cambio fuera de la versión disponible en ese momento, como se aprecia en el paso #3 de la figura 7. Y con esto queda reconstruida la línea base seleccionada.

Figura 6: Reconstrucción #1

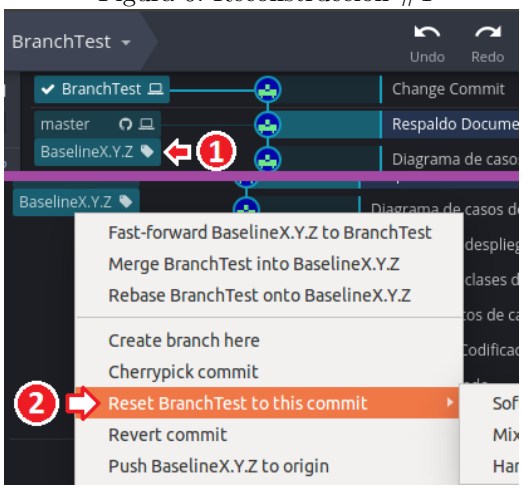
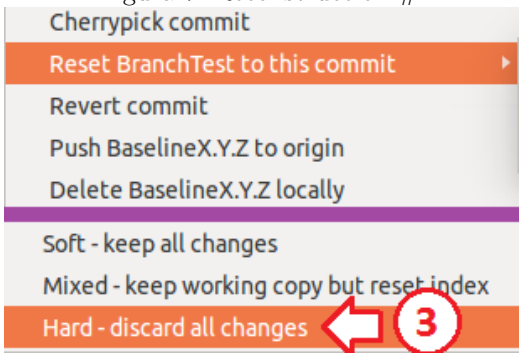


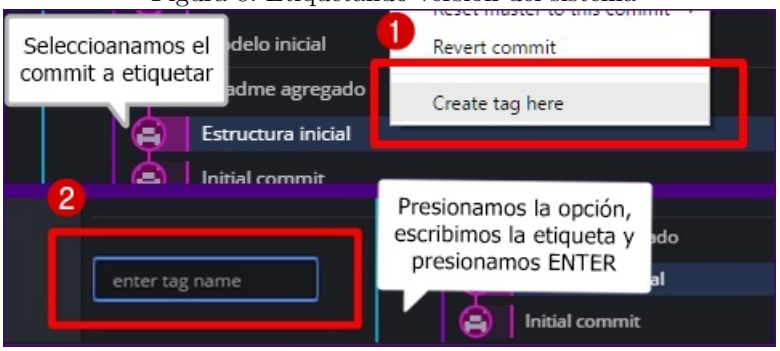
Figura 7: Reconstrucción #2



2.3.4. Tutorial para el etiquetado de versiones

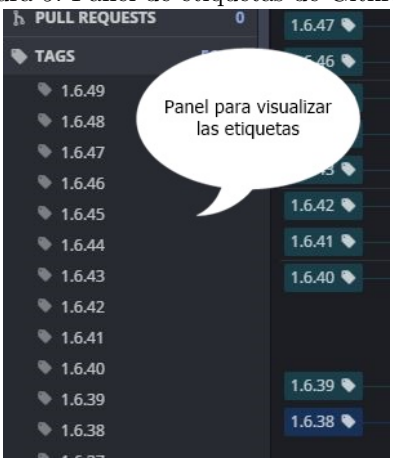
Las etiquetas (*tags*) son punteros estáticos a una linea base en particular. Para etiquetar una versión del sistema se debe seleccionar uno de los *commits* (el que se desea etiquetar) y abrir el menú contextual con *click* derecho, luego presionamos la opción *Create tag here*. Esto nos dirigirá a una caja de texto donde debemos ingresar el nombre de la etiqueta. Es responsabilidad de los desarrolladores seguir el estándar de versionamiento semántico. Ver figura 8.

Figura 8: Etiquetando versión del sistema



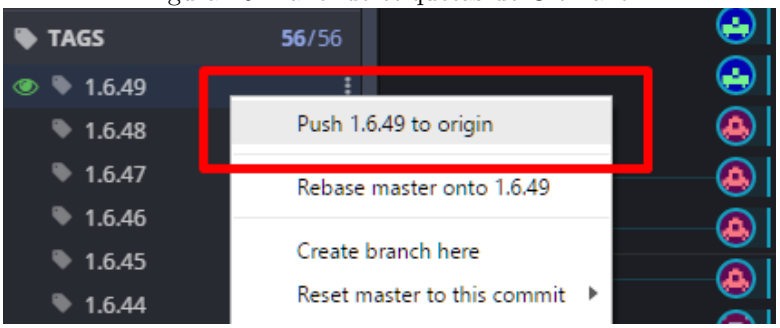
La lista de todas las etiquetas que se han creado puede ser vista en el panel izquierdo del programa. Ver figura 9. Para acceder a un punto etiquetado basta con hacer doble *click* sobre la etiqueta, esto nos redireccionará automáticamente.

Figura 9: Panel de etiquetas de Gitkraken



Las etiquetas pueden ser creadas localmente, y si se desea, se puede hacer que dicha etiqueta este disponible de forma remota haciendo *click* derecho sobre la etiqueta y seleccionando *push tag to origin*. Ver figura 10.

Figura 10: Panel de etiquetas de Gitkraken



Además, se puede incluir anotaciones en las etiquetas con el propósito de añadirles valor adicional. Se puede crear este tipo de anotaciones haciendo *click* derecho sobre la etiqueta y seleccionando la opción *Annotate tag*. Las anotaciones de una etiqueta serán mostradas cada vez que el cursor pase por encima de esta.

3. Control de la Configuración para Procedimientos de Cambio

3.1. Ítems de análisis de requerimientos

3.1.1. Identificación y Documentación de la Necesidad de Cambio

Información de cambio	**Ítem de análisis de requerimientos**
Nombre del ítem:	Identificador distintivo del documento a modificar, esto según la tabla de ítems de configuración del software.
Tipo de cambio:	Eliminación, corrección de error en redacción o modificación del requerimiento.
Fecha de realización:	Enmarca la fecha exacta en la que se propone el cambio
Motivo(s):	Lista de razones por las que se solicita o realiza el cambio.
Autores del cambio:	Nombres y firmas de los responsables de proponer el cambio
Encargados de la evaluación:	Nombre y firma de los encargados de evaluar el cambio
Consideraciones:	Información relevante acerca del requerimiento.
Prioridad:	Brinda un estimado de la urgencia con la que se debe atender la solicitud de cambio

Cuadro 2: Plantilla de cambio para ítem de análisis de requerimiento.

3.1.2. Evaluación y aprobación del cambio

Ante un eventual cambio en un ítem de análisis de requerimientos se define el siguiente procedimiento para su evaluación y posterior aceptación en caso de cumplir con los estándares que se hayan especificado para el tipo de ítem.

1. Debe completarse el documento o plantilla de "información de cambio" para cada uno de los ítems en cuestión. Los datos contenidos dentro de dicho documento deben permitir a los encargados de la aceptación del cambio, el comprender las circunstancias y consecuencias del cambio propuesto.
2. La solicitud de cambio es entregada o notificada a los encargados del control de versiones. Por la naturaleza del proyecto el rol de encargado de aceptación para cambios sobre este tipo de ítem, recae en el scrum master o administrador del proyecto actual según sea el caso, y en ocasiones en el representante del ente externo o sponsor.
3. Se hace disponible el cambio a los miembros del equipo de aceptación de cambios, para su revisión.
4. Los encargados de la aceptación deben revisar que los cambios especificados sean correctos y cumplan con el formato del documento y estándares del mismo.
5. En caso de rechazarse la solicitud de cambio, se notifica al miembro solicitante junto con la información pertinente a los motivos del rechazo.

3.1.3. Implementación de los cambios

En caso que el cambio sea aprobado se define el procedimiento a seguir para su implementación. Además, se debe registrar los encargados de dar el visto bueno y por supuesto notificar a todos los involucrados del cambio e impacto que implica el mismo en las diferentes partes del proyecto.

1. En caso de ser un cambio en los requerimientos del sistema, todo el equipo de desarrollo debe ser notificado sobre el mismo.
2. Se procede a hacer oficial el cambio y poner a disponibilidad de todo el equipo el ítem actualizado.
3. Se definen responsables del acarreo del cambio
4. Por último, se le asigna una prioridad de ejecución

3.2. Ítems de diseño

3.2.1. Identificación y documentación de la necesidad de cambio

Información de cambio	**Ítem de diseño**
Nombre del ítem:	Identificador distintivo del diagrama a modificar.
Fecha de realización:	Enmarca la fecha exacta en la que se propone el cambio
Motivo(s):	Lista de razones por las que se solicita o realiza el cambio.
Aspectos a cambiar:	Detalle del cambio que requiere el diagrama.
Tipo de cambio:	Corrección de errores, rediseño de la solución.
Autores del cambio:	Nombres y firmas de los responsables de proponer el cambio.
Encargados de la evaluación:	Nombre y firma de los encargados de evaluar el cambio
Consideraciones:	Implicaciones o consecuencias de la aplicación del cambio.
Prioridad:	Brinda un estimado de la urgencia con la que se debe atender la solicitud de cambio

Cuadro 3: Plantilla de cambio para ítem de diseño.

3.2.2. Evaluación y aprobación del cambio

Ante un eventual cambio en un ítem de análisis de diseño se define el siguiente procedimiento para su evaluación y posterior aceptación en caso de cumplir con los estándares que se hayan especificado para el tipo de ítem.

1. Debe completarse el documento o plantilla de "información de cambio" para cada uno de los ítems en cuestión. Los datos contenidos dentro de dicho documento deben permitir a los encargados de la aceptación del cambio, el comprender las circunstancias y consecuencias del cambio propuesto.

2. La solicitud de cambio es entregada o notificada a los encargados del control de versiones. Los cambios sobre este tipo de ítem tienen como encargados de revisión al arquitecto de software correspondiente a la sección de diseño específica a modificarse, así como el scrum master o administrador del proyecto según sea el caso para evitar incoherencias entre criterios.

3. Se hace disponible el cambio a los miembros del equipo de aceptación de cambios, para su revisión así como al equipo de desarrollo para confirmar la capacidad del mismo de llevar a implementación el rediseño especificado.

4. Los encargados de la aceptación deben revisar que los cambios realizados sean congruentes con los requerimientos del sistema.

5. Los cambios al diseño deben ser revisados para constatar que se ajusten a los principios de buen diseño especificados para el proyecto.

6. En caso de rechazarse la solicitud de cambio, se notifica al miembro solicitante junto con la información pertinente a los motivos del rechazo.

7. Ante la eventualidad de la necesidad de cambio del diseño bajo una propuesta pobre de rediseño se procede a definir una reunión con el arquitecto de software para llevar a cabo un análisis sobre la situación.

3.2.3. Implementación de los cambios

En caso que el cambio sea aprobado se define el procedimiento a seguir para su implementación.

1. El arquitecto de software aplica las correcciones o adaptaciones que considere necesarias a la propuesta de cambio en caso de no ser él mismo el origen de dicha solicitud.

2. Se adecúa la propuesta a los estándares de calidad de diseño definidos para el sistema.

3. Se actualizan los ítems de configuración correspondientes y se hacen disponibles a todo el equipo mediante la herramienta de control de versiones definida.

4. Se asigna una prioridad de ejecución para el cambio en cuestión

3.3. Ítems de implementación

3.3.1. Identificación y documentación de la necesidad de cambio

Información de cambio	**Ítem de implementación**
Nombre del ítem:	Identificador distintivo de los archivos por modificar.
Tipo de cambios:	Adición de funcionalidad, corrección de errores, optimización.
Fecha de realización:	Enmarca la fecha exacta en la que se propone el cambio
Autores del cambio:	Nombres y firmas de los responsables de proponer el cambio.
Encargados de la evaluación:	Nombre y firma de los encargados de evaluar el cambio
Archivos afectados:	Listado de los archivos fuente afectados de forma indirecta.
Consideraciones:	Implicaciones o consecuencias de la aplicación del cambio.
Prioridad:	Brinda un estimado de la urgencia con la que se debe atender la solicitud de cambio

Cuadro 4: Plantilla de cambio para ítem de implementación.

3.3.2. Evaluación y aprobación del cambio

Ante un eventual cambio en un ítem de implementación se define el siguiente procedimiento para su evaluación y posterior aceptación en caso de cumplir con los estándares que se hayan especificado para el tipo de ítem.

1. Debe completarse el documento o plantilla de "información de cambio" para cada uno de los ítems en cuestión. Los datos contenidos dentro de dicho documento deben permitir a los encargados de la aceptación del cambio, el comprender las circunstancias y consecuencias del cambio propuesto.

2. Por la naturaleza del tipo de ítem, es posible que se requieran cambios en ítems de pruebas relacionados a la implementación cambiaba, en ese caso es necesario también seguir el procedimiento de control de cambios para dichas pruebas. Por ende a la hora de aprobar el cambio, se debería analizar el grado de impacto que el mismo conlleva, pues podría estar fuertemente enlazado a otros componentes que ya estén funcionando a la perfección.

3. La solicitud de cambio es entregada o notificada a los encargados del control de versiones. Los cambios sobre este tipo de ítem tienen como encargados de revisión al scrum master o administrador del proyecto, además se notifica a todos los desarrolladores sobre el cambio ya sea que afecte su área de forma directa o indirecta, aunque estos no participan en el proceso de aprobación.

4. Se hace disponible el cambio a los miembros del equipo de aceptación de cambios para su revisión.

5. Deben hacerse disponibles las pruebas y sus resultados correspondientes para demostrar el correcto funcionamiento del código luego de haber realizado los posibles cambios.

6. El equipo de aceptación revisa las pruebas y resultados proporcionados, junto con el documento de información del cambio para determinar si es aceptable bajo las circunstancias presentadas.

7. En caso de rechazarse el cambio, se notifica al encargado del mismo, junto con las razones detrás de la decisión.

3.3.3. Implementación de los cambios

En caso que el cambio sea aprobado se define el procedimiento a seguir para su implementación.

1. Los cambios en el código se adecúan según los estándares establecidos para la codificación al inicio del proyecto.

2. El encargado de los cambios debe proceder a implementarlos y agregar las pruebas correspondientes a disposición de todos los miembros del equipo.

3. Habiendo siendo ejecutadas las pruebas con resultados exitosos luego de la integración se define el sistema modificado como la última versión del mismo, según el estándar de versionamiento que se ha definido para el proyecto.

Herramientas de aseguramiento

Entre las herramientas que permiten el aseguramiento de calidad durante cambios en ítems de implementación se encuentran:

1. Pylint. [5]
   -Es un software de revisión de código, errores y calidad. Basado en la guía de estilo de Python PEP8.
   -Revisa aspectos como largo de líneas de código, nombres de variables según el estándar definido, verificación de que las interfaces sean implementadas.
   -Aplica para la integración continua al trabajar de forma automatizada con herramientas como Jenkins.

2. Checkmarx SAST [2]
   -Software de análisis de vulnerabilidades de seguridad.
   -Asegura el cumplimiento de estándares de seguridad en los proyectos.
   -Incluye regulaciones de cumplimiento de normas de industria.
   -Trabaja con servidores como Jenkins para integración continua.

3. Codacy [3]
   -Herramientas para revisión del estilo del código.
   -Aspectos de seguridad, duplicación, complejidad y métricas de cobertura.
   -Puede trabajar con repositorios de Git para la verificación de cada commit y push request durante el proceso de cambio.

4. SideCI [7]
   -Herramienta automatizada para revisión del código para la plataforma de GitHub.
   -Revisa aspectos como violación de estilos de código, calidad del mismo, seguridad y dependencias.
   -Los resultados de los análisis son incluidos como comentarios en las solicitudes de *Pull* durante el proceso de cambio.

3.4. Ítems de pruebas

3.4.1. Identificación y documentación de la necesidad de cambio

Información de cambio	**Ítem de implementación**
Nombre del ítem:	Identificador distintivo de las pruebas por modificar.
Motivo(s):	Lista de razones por las que se solicita o realiza el cambio.
Aspectos a cambiar:	Detalle de los módulos que serán cambiados.
Autores del cambio:	Nombres y firmas de los responsables de proponer el cambio.
Encargados de la evaluación:	Nombre y firma de los encargados de evaluar el cambio
Fecha de realización:	Enmarca la fecha exacta en la que se propone el cambio
Posibles afectaciones:	Describe posibles escenarios que se pueden presentar al llevar a cabo el cambio propuesto
Prioridad:	Brinda un estimado de la urgencia con la que se debe atender la solicitud de cambio

Cuadro 5: Plantilla de cambio para ítem de pruebas.

3.4.2. Evaluación y aprobación del cambio

Ante un eventual cambio en un ítem de pruebas se define el siguiente procedimiento para su evaluación y posterior aceptación en caso de cumplir con los estándares que se hayan especificado para el tipo de ítem.

1. Debe completarse el documento o plantilla de "información de cambio" para cada uno de los ítems en cuestión. Los datos contenidos dentro de dicho documento deben permitir a los encargados de la aceptación del cambio, el comprender las circunstancias y consecuencias del cambio propuesto.

2. La solicitud de cambio es entregada y notificada a los encargados del control de versiones. Los cambios sobre este tipo de ítem tienen como encargados de revisión al equipo de desarrollo de los que se esperan sus criterios propios.

3. Se hace disponible el cambio a los miembros del equipo encargado de la revisión de cambios.

4. Se ejecutan las pruebas resultantes del cambio para comprobar su correctitud, dichos resultados son analizados por los encargados de desarrollo en el área de la prueba.

5. En caso de rechazo de los cambios, se descartan las pruebas y se notifica al generador de la solicitud de la necesidad de replantear dichos cambios.

3.4.3. Implementación de los cambios

En caso que el cambio sea aprobado se define el procedimiento a seguir para su implementación.

1. Si el cambio en el ítem de pruebas es aceptado, se estandariza según las especificaciones definidas para la codificación en caso de necesitarlo.

2. Se agregan los cambios implementados a la línea base del sistema para disponibilidad de todos los miembros del equipo.

3. Se asigna un responsable del acarreo del cambio

4. Por último, se deberá establecer una prioridad de implementación.

Herramientas de aseguramiento

Entre las herramientas que permiten el aseguramiento de calidad durante cambios en ítems de implementación se encuentran:

1. SonarQube [8]
   -Plataforma de inspección continua para revisión de calidad de código.
   -Sus funcionalidades son amplias, pero destaca la generación de reportes sobre purbeas unitarias y estándares de código para las mismas.
   -Permite la integración con herramientas como Jenkins para construcción automatizada de proyectos.

Referencias

[1] Calderón, S. Trejos, I. (2017). *Actividades de Aseguramiento de la Calidad*. Recuperado de Carpeta del Curso, archivo ACS2_actividades

[2] Checkmarx Official Website.(2017). *Checkmarx SAST*. Recuperado de `https://www.checkmarx.com/technology/static-code-analysis-sca/`

[3] Codacy Official Website.(2017). *Automated code reviews and analytics*. Recuperado de `https://www.codacy.com`

[4] Hart S. (2011). *PM Foundations–The Change Control Process*. Recuperado de `https://pm-foundations.com/2011/06/16/pm-foundations-%E2%80%93-the-change-control-process/`

[5] Pylint Official Website.(2017). *Pylint*. Recuperado de `https://www.pylint.org`

[6] Sanz Masedo, R. (2014). *Monitoreo y Control del Proyecto*. Recuperado de `http://www.uv-mdap.com/programa-desarrollado/bloque-i-el-ciclo-de-vida-del-proyectos/monitoreo-y-control-del-proyecto/`

[7] SideCI Official Website.(2017). *CI for Automated Code Review*. Recuperado de `https://sideci.com`

[8] SonarQube Official Website.(2017). *CI for Automated Code Review*. Recuperado de `https://sideci.com`

[9] Tom Preston-Werner, cofounder of GitHub.(2017). *Semantic Versioning 2.0.0*. Recuperado de `http://semver.org/`