Estrategia y roadmap de calidad y verificación

PIS, Grupo 10

Facultad de Ingeniería Udelar

Gestión de versiones:

Gestión de versiones de documentos

Todos los documentos se encuentran con sus respectivas últimas versiones en un [repositorio de github](https://github.com/TiberiusGG/baseline-PIS) llamado baseline-PIS, de esta forma podemos fácilmente gestionar que documentos fueron modificados en que momento.  
Este repositorio se nutre de otra carpeta llamada linea base que se encuentra en nuestro drive donde se encuentran los documentos aun sin terminar de construir/iterar.

Tenemos en la línea base todos los roadmaps y planes tanto de configuracion, gestion de proyecto y calidad/verificación. Además tenemos el documento de riesgos, guias de configuración del entorno de trabajo, DoD, DoR.  
  
Para iterar un documento el procedimiento es el siguiente:

1. Si el documento a modificar fue entregado en otra iteración se crea una copia de este ultimo en drive y se modifica esta copia, indicando la versión con el siguiente formato vX.Y donde X indica cambios grandes e Y arreglos menores.  
   Si el documento se crea nuevo, empieza en la versión v1.
2. Tras haber modificado el documento y haber sido validado, se suma a las releases-notes los cambios realizados en este documento
3. Se avisa al responsable de SCM de estos cambios de forma que pueda descargar desde el drive los documentos cambiados e integrarlos en el repositorio de línea base para preparar la release de la siguiente iteración.

Gestión de versiones de código

Para la gestión de versiones, contamos con dos repositorios remotos en GitHub donde decidimos separar el repo de Frontend con el de Backend para distribuir la cantidad de desarrolladores entre ambos proyectos e independizar las dos ramas de desarrollo.  
  
En ambos repositorios se tendrán dos ramas clave:

1. Una rama **master** donde estará la última versión del producto en producción.
2. Una rama **develop** donde los desarrolladores insertarán las nuevas funcionalidades mediante Pull Request, las cuales como se comenta en el roadmap de verificacion y calidad debe contar con al menos una aprobación.

Además utilizaremos un estándar de manejo de versiones bien conocido llamado **GitFlow** el que establece un workflow o flujo de trabajo con el que se realiza el control de inserciones de código y versiones de la aplicación.  
  
La documentación de GitFlow aparece más detallada en la sección de recursos de este documento, pero de forma general los desarrolladores solo podrán hacer PRs a la rama develop a través de ramas de feature o bugfix y la única forma de insertar cambios a master es a través de hotfix o de una release desde develop que integre todas las funcionalidades de la iteración.

El encargado de crear la rama de release es el responsable de SCM, este creará un Pull Request desde develop con determinada versión antes de finalizar cada iteración, cualquier cambio que se quiera realizar sobre la release solo será mediante bugfixes y no se deberían incorporar nuevas funcionalidades (features) a ella.  
Cuando todo el equipo de desarrollo y el cliente de el OK sobre los cambios de la release, esta rama se fusionará con master y con develop (por los fixes que se podrían haber incorporado a la release) dejando preparado el ambiente para la siguiente release en la siguiente iteración.

Gestión de ambientes y despliegue:

Heroku es una plataforma como servicio en la nube que permite subir y ejecutar aplicaciones enteramente en la web, soporta múltiples lenguajes de programación, incluyendo los de nuestro interés, ruby y javascript.

Además maneja apps con los distintos desarrollos hosteados en la plataforma, para este proyecto se usarían 2 apps, una ejecutando el front-end basado en react y otra ejecutando el back-end basado en ruby.

Los deploy sobre Heroku, independientemente de la app, se realizan en base al repositorio git explicado anteriormente (el responsable de hacerlos es quien se responsabiliza del git) y se contarán con 2 ambientes dentro de cada app:

* Staging: el cual es un ambiente de pruebas similar al ambiente de producción, en este se harán deploy sólo desde la rama release, cuando se hagan estos deploys, el responsable avisará a todos los desarrolladores involucrados para que pruebe que sus desarrollos funcionen correctamente y que no se hayan generado conflictos con los demás desarrollos de la release en cuestión. Luego de que cada desarrollador aseguró que todo funciona correctamente, el responsable hará una última revisión de aceptación con el cliente y la pasa al ambiente de producción.
* Producción: el cual es el ambiente donde corre la versión de la WebApp sobre la cual trabaja el cliente. En este ambiente solo se hacen deploys desde la rama master del git y sólo se deberían hacer deploys luego de una release aprobada por el cliente o de un hotfix.

Aún se está analizando cuál es la manera más eficiente de manejar las variables de configuración para cada ambiente, por ejemplo, para el frontend sería la URL de la app de Heroku que tiene la api desarrollada en ruby, la misma sería distinta entre el ambiente de staging y el de producción.

¿Cuándo sucederán estos sucesos?

En el proceso de iteración de las versiones de código o documentos es importante que se sepa cuando se van a terminar las actividades asociadas a cada funcionalidad o entregable y que los integrantes sepan que se terminaron, por esto, es necesario un mecanismo para que todos estén al tanto de estos sucesos.  
Con objetivo de promover el conocimiento general del estado actual del proyecto se realizan reuniones semanales de sincronización, donde cada integrante cuenta que tareas terminó y si no las terminó en qué estado se encuentran.  
Además contamos con la herramienta de Jira donde podemos ver el tiempo de finalización de cada tarea clickeando sobre cada card. Estamos buscando una forma de facilitar la visualización de los ETA de cada actividad para no tener que buscar incidencia por incidencia.

En cuanto los deploys, estos se realizarán en un entorno de prueba el sábado anterior a finalizar el sprint así dar tiempo al domingo para realizar los fixes pertinentes, finalmente se realizará el deploy del código y entrega de documentos en sus últimas versiones.

Recursos

Los desarrolladores deben estar familiarizados con la documentación de [GitFlow](https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow). El responsable de deploy también debe contar con la documentación de [Heroku](https://help.heroku.com/).

Para el código se utilizarán dos repositorios en GitHub, uno para el back-end y otro para el front-end.

Para documentos de línea base también habrá un repositorio.

En cuanto a documentación de la configuración se manejan canales en Discord para anunciar los pull request, un canal para back-end y otro para front-end.

Para estar informados del estado actual de cada tarea se utiliza un tablero de [Jira](https://fing.atlassian.net/jira/software/projects/PIS2021/settings/issuetypes/10002)