Estrategia y roadmap de calidad y verificación

PIS, Grupo 10

Facultad de Ingeniería Udelar

Introducción  
En el presente documento se busca especificar cómo se realizará el abordaje de la configuración de artefactos del proyecto, se tratarán dos secciones separadas, la que refiere a la gestión de versiones mediante repositorios de GitHub y la que refiere a gestión de ambientes y despliegue de las diferentes liberaciones del producto en Heroku.

Gestión de versiones:

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.

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.

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.