Descripción

La actividad es individual, por lo que cada uno va a tener el rol de desarrollador y debe abordar la tarea de forma independiente desde el inicio hasta el final. La unión de todas las tareas podría ser el producto final que un cliente ha podido solicitar. En un proyecto real, lo normal es tener una tarea asignada y responsabilizarse de ella de principio a fin.

A continuación, se especifican los estados por lo que puede pasar un issue dependiendo del objetivo para el que se creen.

En el tablero del proyecto de cada grupo se han creado dos nuevas columnas quedando de la siguiente forma:

- Backlog: issues pendientes de analizar por el equipo de desarrollo, estimar y priorizar
- To Do: issues ordenadas por prioridad y listas para empezar a desarrollar
- In Progress: issues en desarrollo
- QA (Quality Assurance): issues disponibles para pruebas
- Done: finalizada (este estado puede utilizarse para issues que no requieren desarrollo y se finalizan directamente como tareas de análisis o como cerrado el desarrollo pero antes de pasar a las pruebas de usuario)

En un proyecto real, podría haber mínimo dos columnas como:

- Ready to deploy: está probado a la espera de implantar
- Implemented: implantado en entornos productivos

Dado que cada desarrollador va a tener una tarea asignada que podría corresponder a una funcionalidad del proyecto, se deben seguir unas convenciones de nomenclatura de archivos, carpetas, ramas, commits y pull request. En el Anexo C se puede ver la definición.

En cada carpeta del repositorio donde se vaya a subir algún archivo se debe crear una carpeta dentro nombrada con el usuario. Ejemplo, si mi usuario es NFM-git y voy a subir un archivo a src el árbol de carpetas queda:

/src/NFM-git/archivo.md

Pasos para seguir el ciclo de vida de un issue en GitHub Projects

Paso de Backlog a To Do

El issue asignado ya se encuentra en la columna To Do para poder empezar a desarrollar.

Paso de To Do a In Progress

Cuando el desarrollador comience la tarea, debe moverla a la columna In progress y mencionar al usuario NFM-git incluyendo un comentario explicando lo que se va a realizar. Si no se ha practicado con esta funcionalidad en el Anexo A se puede ver como hacerlo.

Cada tarea se debe desarrollar en una rama propia creada desde la rama principal (main). Antes de empezar a desarrollar, se debe crear la rama y asegurarse que se está posicionado en esa rama durante todo el desarrollo. Se recomienda hacer un commit por cada cambio para mantener la trazabilidad. Si no se ha trabajado anteriormente con ramas independientes, en el Anexo B se especifica como podar hacer cada paso.

El desarrollo consiste en crear una sencilla aplicación (se puede utilizar cualquiera ya desarrollada en el módulo de programación). Durante el desarrollo, se recomienda hacer commit para guardar los cambios. Una vez finalizado el desarrollo, los archivos deben subirse al repositorio propio en las carpetas correspondientes según su tipo, ya sea usando GitHub o GitHub Desktop. Si se necesitan nuevas carpetas, estas también deben crearse en la rama propia. Solo cuando se haga un merge con la rama principal, los cambios (carpetas y archivos) aparecerán en el repositorio.

Es muy importante mantener actualizados los readme de las carpetas que corresponda, seguir la estructura de carpetas y las convenciones de nomenclatura del proyecto indicadas en la descripción de la actividad.

Con esto la tarea está preparada para pasar al siguiente estado.

Paso de In Progress a QA

Cuando el desarrollador cambie de estado la tarea a QA debe mencionar al usuario NFM-git incluyendo un comentario explicando lo que se va a realizar. Para realizar las pruebas se deben definir y ejecutar las pruebas de caja blanca y caja negra. El documento resultante de las pruebas se debe dejar en la carpeta /docs/ siguiendo el mismo procedimiento que para subir el desarrollo en el paso anterior (subir al repositorio y commit de los cambios).

Es muy importante mantener actualizados los readme de las carpetas que corresponda, seguir la estructura de carpetas y las convenciones de nomenclatura del proyecto indicadas en la descripción de la actividad.

Con esto la tarea está preparada para pasar al siguiente estado.

Paso de QA a Done

Antes de pasar la tarea a Done, se debe crear una pull request para solicitar integrar los cambios de la rama de trabajo en la rama principal. Normalmente esta pull request le llega a un responsable del proyecto que revisa y autoriza la fusión de las ramas, pero en este caso no va a ser necesario y se podría hacer el merge directamente. Con esto quedan consolidados los cambios en la rama principal. Si no se ha trabajado anteriormente con pull request y merge, en el Anexo B se especifica como podar hacer cada paso.

Cuando el desarrollador cambie de estado la tarea a la columna Done y mencionar al usuario NFM-git incluyendo un comentario explicando lo que se va a realizar.

Es muy importante mantener actualizados los readme de las carpetas que corresponda, seguir la estructura de carpetas y las convenciones de nomenclatura del proyecto indicadas en la descripción de la actividad.

Anexos

Anexo A Mencionar a un usuario en un comentario de un issue

- Mencionar a uno o varios usuarios en la creación de una tarea

Para mencionar o notificar a un compañero del proyecto sobre dudas, aclaraciones, comentarios de desarrollo o cambios de estado:

- En la descripción del issue o en un comentario, se escribe @ seguido del nombre de usuario
- Aparece la lista de compañeros que puedes seleccionar
- Se puede notificar a uno o varios compañeros, incluyendo @ antes de cada nombre

Anexo B Crear nueva rama, commit, merge y pull request

A continuación se indican los pasos para realizar cada acción aunque también se puede tener como referencia la documentación que proporciona GitHub.

- Enlace del manual de uso de GitHub: https://docs.github.com
- Enlace del manual de uso de GitHub Desktop: https://docs.github.com/es/desktop

Crear una rama desde la rama principal

- En GitHub (web)
  - Abrir el repositorio en GitHub
  - Comprobar que se está en la rama principal (`main`)
  - Acceder al apartado 'Branches' pulsar 'New branch'. Aparece un modal para introducir la información necesaria para crear la nueva rama, siguiendo las convenciones de nomenclatura y seleccionando la rama `main` del desplegable que aparece
  - Pulsar 'Create new branch' para la rama propia
- En GitHub Desktop
  - Abrir GitHub Desktop
  - Comprobar que se está en la rama principal (`main`)
  - En el menú 'Branch' seleccionar 'New Branch'. Aparece un modal para introducir la información necesaria para crear la nueva rama siguiendo las convenciones de nomenclatura y comprobando que se crea desde la rama `main`
  - Pulsar 'Create new branch' para crear la rama propia

Hacer commit en la rama propia

Antes de hacer el commit hay que asegurarse de estar en la rama propia. El nombre de la rama en la que se está posicionado aparece en la parte superior izquierda del listado de archivos. Si no se está en la rama propia, se debe seleccionar del desplegable de ramas.

- En GitHub (web)
  - Abrir el repositorio en GitHub
  - Realizar los cambios y pulsar 'Commit changes' para guardar los cambios introduciendo la información necesaria para hacer el commit siguiendo las convenciones de nomenclatura
- En GitHub Desktop

    - Abrir GitHub Desktop
    - Realizar los cambios que posteriormente se ven en la pestaña 'Changes' y rellenar un mensaje en 'Summary' descriptivo
    - Pulsar 'Commit to nombre_rama_propia' en la parte inferior izquierda para confirmar el commit introduciendo la información necesaria para crear la nueva rama siguiendo las convenciones de nomenclatura
    - Después, opcionalmente, se puede pulsar en 'Push origin' para subir los commit del repositorio local al repositorio remoto en GitHub

Pull request (PR)

- En GitHub (web)
    - Abrir el repositorio en GitHub
    - Comprobar que se está en la rama propia
    - Crear el PR pulsando 'New Pull Request'
        - Comprobar que se va a comparar la rama `main` con la rama propia
        - o
        - Pulsar 'Compare & pull request' si aparece directamente un mensaje en amarillo
    - Introducir la información necesaria para crear la PR siguiendo las convenciones de nomenclatura
    - Pulsar 'Create pull request' para dar de alta la PR
- En GitHub Desktop
    - Abrir GitHub Desktop
    - Comprobar que se está en la rama propia
    - Comprobar haber hecho 'Push origin' para subir los commits que están en el repositorio local al repositorio remoto en GitHub
    - Crear el PR pulsando 'Create Pull Request' donde se abre el navegador con GitHub (comprobar que se va a comparar la rama `main` con la rama propia) e introducir la información necesaria para crear la PR siguiendo las convenciones de nomenclatura
    - Pulsar 'Create pull request' para dar de alta la PR

Merge

El merge se hace desde la Pull Request (PR).

- En GitHub
    - Abrir el repositorio en GitHub
    - Seleccionar la PR que se va a integrar desde la pestaña 'Pull request'
    - Comprobar que no hay conflictos antes de la integración porque si no hay que resolverlos antes
    - Seleccionar ' Merge pull request' e introducir la información necesaria para hacer el merge siguiendo las convenciones de nomenclatura
    - Pulsar 'Confirm merge' para finalizar el proceso y poder ver los cambios en la rama `main`
- En GitHub Desktop
    - Realizar el merge desde GitHub según las especificaciones anteriores

    - Después del merge volver a GitHub Desktop
    - Seleccionar la rama `main` y pulsar 'Fetch origin' y después 'Pull origin' para actualizar el repositorio local con los cambios integrados

Anexo C Convenciones de nomenclatura

- Carpetas y archivos
    - Nombrar en minúsculas
    - No usar espacios en el nombre
    - Usar guiones medios (-) o bajos (_)
- Rama
    - Título: usuario-fecha-idtarea-descripcion_breve (ejemplo: NFM-git-20260219-#10-validacion_datos_cliente)
    - Descripción
- Commit y pull request
    - Título: usuario-idtarea-descripcion_breve
    - Descripción