# Tarea 3: Git merge y pull request

## Changelog

|  |
| --- |
| El fichero CHANGELOG.md es un **histórico de los cambios** que se hacen en el proyecto.  El fichero se lee de abajo a arriba, es decir, los cambios más antiguos estarán al final del fichero, y los nuevos cambios estarán al principio.  En la sección “Unreleased” iremos escribiendo los cambios que vayamos haciendo en la rama de trabajo. Justo antes de que los cambios lleguen a la rama “develop” aumentaremos el número de versión y dejaremos vacía la sección “Unreleased”.  **Guía de uso**   * El changelog es para que lo lean los humanos, no para las máquinas * Debería haber una sección para cada versión * Los mismos tipos de cambios se deben agrupar * Las versiones y secciones deben ser enlazables * Las últimas versiones aparecen primero * Se muestra la fecha de entrega de cada versión * Se indica si se sigue o no el Versionado Semántico   **Tipos de cambios**:   * **Added**: para funcionalidades nuevas. * **Changed**: para cambios en la funcionalidad existente. * **Deprecated**: para funcionalidades que pronto van a quedar obsoletas * **Removed**: para funcionalidades eliminadas * **Fixed**: para correcciones de bugs. * **Security**: en caso de vulnerabilidades de seguridad.   **Versionado Semántico:**  Dado un número de versión MAJOR.MINOR.PATCH, incrementar:   * Versión **MAJOR** cuando se hacen cambios en las APIs que provocan **incompatibilidades** con las versiones anteriores, * Versión **MINOR** cuando los cambios que se hacen son compatibles hacia atrás, * Versión **PATCH** cuando se corrigen bugs de forma compatible hacia atrás.   Referencias:   * <https://keepachangelog.com/en/1.1.0/> * <https://semver.org/> |

* 1. Descárgate una copia de la versión del repositorio remoto que creaste en la última clase.
  2. Crea una rama con el nombre ‘develop’
  3. Cambia la rama actual a la rama ‘develop’
  4. Añade un fichero **CHANGELOG.md** en la carpeta raíz del repositorio con el siguiente contenido:

# Changelog

Todos los cambios destacables en este proyecto se documentarán en este fichero.

Este formato está basado en [Keep a Changelog](<https://keepachangelog.com/en/1.0.0/>), y este proyecto se adhiere a [Semantic Versioning](<https://semver.org/spec/v2.0.0.html>).

## [Unreleased]

## [0.2.0] - <PON AQUÍ LA FECHA DE HOY>

### Added

- Fichero CHANGELOG.md

## [0.1.0] - 2021-12-15

### Added

- Ficheros Book.java y Member.java

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* 1. Lleva estos cambios hasta el repositorio remoto.
  2. Obtén pantallazos en Github que demuestren que los cambios han llegado a la rama remota ‘develop’

## Merge

|  |
| --- |
| El comando **git merge** fusionará cualquier cambio que se haya hecho en la base de código en una rama separada de tu rama actual como un nuevo commit.  La sintaxis del comando es la siguiente:  **git merge NOMBRE-DE-LA-RAMA**  Por ejemplo, si estás trabajando actualmente en una rama llamada **develop** y deseas fusionar los nuevos cambios que se hayan realizado en una rama llamada **new-features**, ejecutarías el siguiente comando:  **git merge new-features**  **(estando en la rama develop)** |

* 1. Cambia la rama actual a la rama ‘main’
  2. Fusiona la rama ‘develop’ a la rama ‘main’
  3. Lleva los cambios al repositorio remoto

Pull Request

|  |
| --- |
| Las **solicitudes de incorporación de cambios, (o** **Pull Requests)** son una funcionalidad que facilita la colaboración entre desarrolladores.  Son un mecanismo para que los desarrolladores notifiquen a los miembros de su equipo que han terminado una función. Una vez la rama de función está lista, el desarrollador realiza la solicitud de incorporación de cambios.  Pero es mucho más que una notificación: es un foro especializado para debatir sobre una función propuesta. Si hay algún problema con los cambios, los miembros del equipo pueden publicar feedback en las solicitudes de incorporación de cambios e incluso modificar la función al enviar confirmaciones de seguimiento. El seguimiento de toda esta actividad se realiza directamente desde la solicitud de incorporación de cambios.  Cuando realizas una pull request, lo que haces es solicitar que otro desarrollador (por ejemplo, el mantenedor del proyecto) incorpore (o haga un pull) una rama de tu repositorio al suyo. Por tanto, para realizar esta solicitud, debes proporcionar la siguiente información: **el repositorio de origen, la rama de origen, el repositorio de destino y la rama de destino**. |

Vamos a prohibir merges directos en la rama develop, creando una regla que nos dice que las ramas que contengan el nombre develop necesitan un pull resquest previo a un merge.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

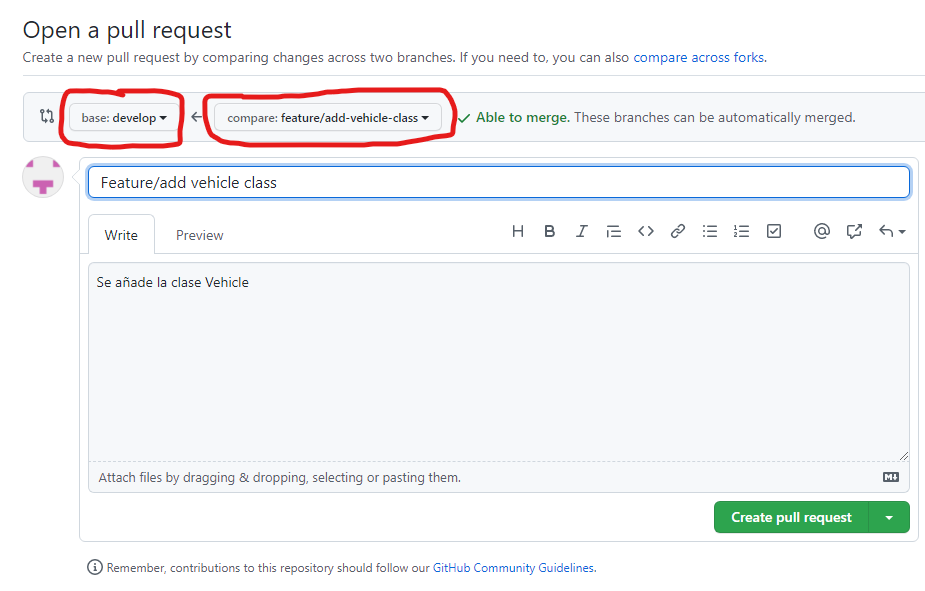
Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

* 1. Cambia la rama actual a la rama ‘develop’
  2. Crea una nueva rama ‘feature/add-vehicle-class’
  3. Cambia a la nueva rama
  4. En la carpeta ‘java’ añade el fichero ‘Vehicle.java’ adjunto a la tarea.
  5. Edita el fichero **Changelog** de acuerdo con los cambios realizados
  6. Comprueba el estado del repositorio
  7. Lleva estos cambios hasta el repositorio remoto con el mensaje “Creación de clase Vehicle”
  8. Cambia la rama actual a la rama ‘develop’
  9. Fusiona la rama feature en la rama develop con el mensaje “Merge de la funcionalidad add vehicle”
  10. Intenta llevar los cambios hasta el repositorio remoto (debe fallar puesto que la rama develop está protegida)
  11. En Github, creamos un Pull Request

Texto

Descripción generada automáticamente con confianza media



Y aparece el “foro” de la pull request creada

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

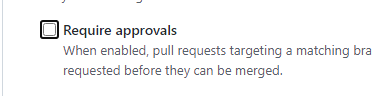
Descripción generada automáticamente

No nos deja, tenemos que cambiar algo de la política de seguridad

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Desmarcar esta opción

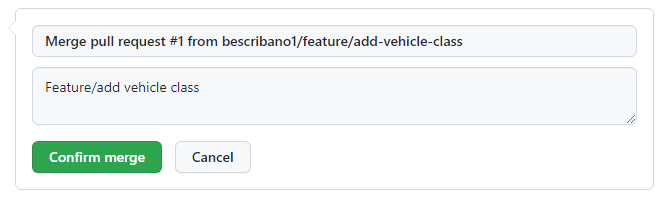


Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* 1. En local, apuntando a la rama develop, sincronizamos el repositorio local con los cambios del repositorio remoto
  2. En local, comprobamos el histórico de cambios de la rama ‘develop’, y vemos que el último cambio es el Merge del pull request

## Gitflow

Lo que hemos hecho en esta práctica forma parte del flujo de trabajo denominado Gitflow.

* 1. Investiga acerca de Gitflow, ¿Qué pasos habría que dar para corregir un bug de una versión que se encuentre en producción?
  2. Investiga otros flujos de trabajo. ¿En qué se diferencia “Gitflow” de “Trunk-based development”?