IMAGE CAMPUS

- OTRA EDUCACIÓN -

Requisitos sobre versionado y repositorio para proyectos

Desarrollo de Videojuegos I - 2023

Sergio Baretto

1) Proyecto versionado

El proyecto debe estar versionado usando el sistema git.

2) Proyecto en GitHub

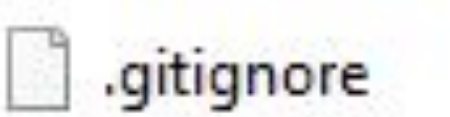
El proyecto debe estar alojado en GitHub.

3) Gitignore correcto. No versionar archivos que se pueden regenerar

El repositorio:

- debe tener un gitignore correcto.
- no debe incluir archivos que se pueden regenerar.

Debe haber un archivo .gitignore que contenga las reglas correctas para ignorar lo que no se debe versionar, que es aquello que se puede regenerar (ejecutable, archivos intermedios de la compilación, etc.)



4) Escribir buenos mensajes de commit

Los mensajes de commit deben ser claros y específicos del cambio que se está realizando.

También deben ser consistentes en cuanto a idioma, mayúsculas/minúsculas, tiempos verbales, etc.

Para más info googlear “how to write a good commit message” o ver por ejemplo:

- https://chris.beams.io/posts/git-commit/
- https://www.freecodecamp.org/news/writing-good-commit-messages-a-practical-guide/

5) Realizar commits atómicos

Un commit debería incluir cambios de una unidad de trabajo sola. Esto implica que debería atacar una funcionalidad o mejora específica y no mezclar cambios relacionados con varios aspectos diferentes.

Por ejemplo, si se implementa la funcionalidad de salto para el personaje, el commit no debería incluir correcciones en el gitignore, cambios en el enemigo, otras mejoras al proyecto, etc., sino solamente el agregado de archivos, cambios en el código y modificaciones al proyecto que fueron necesarios para agregar dicha funcionalidad, dejando a la corrección del gitignore como un commit aparte, al cambio de enemigo como otro, etc.

Continúa

5) Realizar commits atómicos (continuación)

De la misma manera, si se implementa la funcionalidad de salto tampoco es que hay que hacer un commit por cada cambio mínimo. Si se modifican 2 archivos para lograr dicha funcionalidad no es que hay que hacer un commit por cada archivo sino que el commit agrupa a ambos porque implica el agregado de dicha funcionalidad.

En general, el mensaje que ponemos para el commit nos permitirá dar cuenta de si estamos haciendo bien esto, ya que debería ser corto y específico. Si en un momento tenemos que pensarlo demasiado o listar demasiados aspectos es que seguramente no seguimos esta regla de la atomicidad.

Más info: https://www.freshconsulting.com/atomic-commits/

6) Taggear cada nueva versión del juego

Cuando se realizan varios commits y se completa un conjunto de funcionalidades tales que producen un avance en el juego que consideramos marca una nueva versión del mismo (0.1, 0.2, etc.) se debe crear un tag en el commit en que se cerró la nueva versión.

Este tag debe ser el número de versión (0.1, 0.2, etc.) y se debe pushear al repositorio remoto (git push --tags).

Más info: https://www.studytonight.com/git-guide/git-tag

7) Generar release en GitHub que asocie versión del juego a su ejecutable

Por cada versión del juego (0.1, 0.2, etc.) que se haga, luego de generar el tag debe crearse un release de GitHub.

El release debe incluir, como asset para descargar, un archivo comprimido que contenga el build del juego en configuración release, de esa versión taggeada.

Como comentario del release puede escribirse un changelog de lo incorporado en dicha versión.

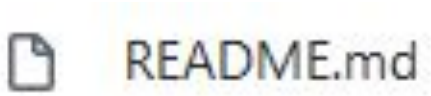
Más info:
https://docs.github.com/es/repositories/releasing-projects-on-github

8) El repositorio debe contar con un archivo README.md

El archivo README.md debe estar en la base del directorio (a la misma altura que el .gitignore).

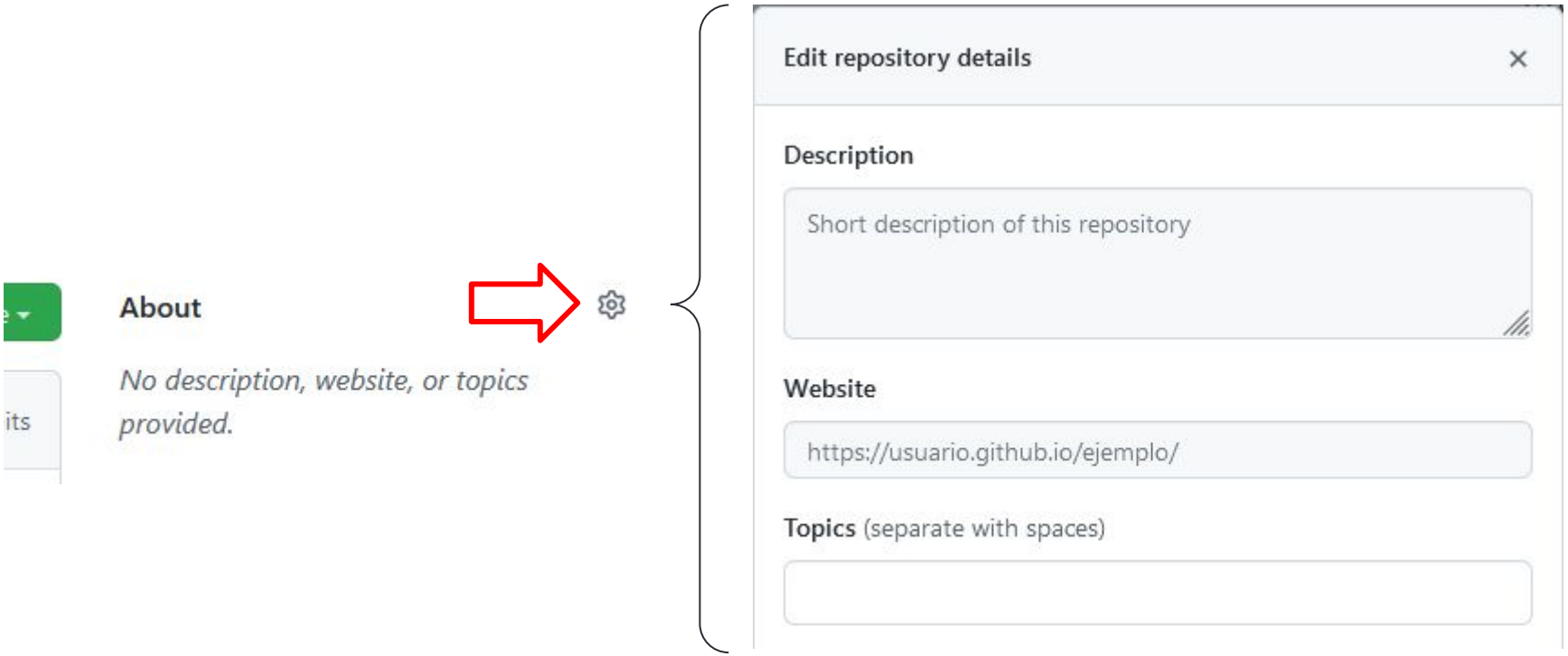
El nombre de este archivo debe ser todo en mayúsculas (no así su extensión): README.md.

Allí se debe escribir como mínimo una descripción del proyecto. Se recomienda además agregar logo con título del juego, algunas screenshots, autor y forma de contactarlo, etc.



9) Ingresar descripción, link y topics

Agregar descripción del proyecto, link a sitio de itch.io donde se aloja el juego (cuando esté publicado) y tópicos (etiquetas).



NOTA: no olvidar los errores comunes mencionados y otros aspectos de buenas prácticas sobre versionado señalados en clase.

Links útiles: algunos recursos mencionados en las diapositivas anteriores:

- https://chris.beams.io/posts/git-commit/
- https://www.freecodecamp.org/news/writing-good-commit-messages-a-practical-guide/
- https://www.freshconsulting.com/atomic-commits/
- https://www.studytonight.com/git-guide/git-tag
- https://docs.github.com/es/repositories/releasing-projects-on-github