Versionamiento semántico

semver.org

v0.	**0.**	**0**
Versión mayor	Versión menor	Versión parche
architecture	feature	bugfix

Reglas de versionamiento

- Nunca utilizar números negativos
- Nunca cambiar el contenido de la versión release, necesita hacerse una nueva versión para taguear el nuevo cambio
- Cuando la versión mayor es cero, se considera un proyecto en desarrollo inestable
- Cuando se soluciona un error se debe incrementar la versión de parche
- Cuando se agrega una nueva funcionalidad que no rompe con la versión actual, se incrementa la versión menor y se regresa la versión de parche a cero
- Cuando se crea un cambio que no es compatible con la versión anterior, se incrementa la versión mayor y a la vez la versión menor y el parche se deben regresar a cero

Ramas

Existen dos tipos de ramas, las que viven por siempre: **main** y **develop**. Las que se crean y eliminan cada que se utilizan: *feature*, *release* y *hotfix*.

Características de las ramas

- *Feature*. Nuevas funcionalidades, actualizaciones y bugfixes encontrados en desarrollo. Nace de la rama develop y al terminar se une a la rama develop. Una vez mergeada debe eliminarse.
- **Develop.** Versión inestable en desarrollo. Se crea al inicio del proyecto, debe estar protegida en el repositorio para que nunca se elimine.
- *Release*. Versión de pruebas candidata a salir como siguiente versión de producción. Nace de develop y se une con main al validarse. Una vez mergeada debe eliminarse.
- *Hotfix*. Bugfix de errores que se encontraron en producción. Nace de main y al solucionarse se debe unir tanto a main como a develop. Una vez mergeada debe eliminarse.
- **Main.** Versión estable en producción. Se crea al inicio del proyecto, debe estar protegida en el repositorio para que nunca se elimine.

Flujo de las ramas y su versionamiento

Para versionar tanto en tag como en el package.json utilizar los comandos de `npm version` en la terminal

- `npm version patch` incrementa el último dígito de la versión, crea un tag y lo publica
- `npm version minor` incrementa el dígito de en medio de la versión, crea un tag y lo publica
- `npm version major` incrementa el primer dígito de la versión, crea un tag y lo publica

Para fijar las versiones que se utilizan en el package y no estar actualizando el archivo package-lock.json se puede utilizar el comando `npm config set save-exact=true`

feature	**develop**	*release*	*hotfix*	**main**
checkout feature/funcionalidad				
	merge feature a			

	develop			
delete feature branch	npm version minor			
checkout feature/funcionalidad-2				
checkout feature/bugfix-algo-por-arreglar				
	merge bugfix a develop			
delete feature/bugfix branch	npm version patch			
	merge feature2 a develop			
delete feature2 branch	npm version minor	checkout release candidate v1.0.0-rc.1		
feature/actualizacion-datos				
	merge actualizacion a develop			
delete feature/actualizacion branch	npm version minor	merge release candidate v1.0.0-rc.2		
				merge release a main
		delete release branch		npm version major git push --tags
	merge main a develop			
			checkout hotfix/un-error-a	

			-solucionar	
				merge hotfix a main
			delete hotfix branch	npm version patch git push --tags
	merge main a develop			

Github
https://github.com/flkt-crnpio/semantic-version-npm-gitflow

Ref
https://www.freecodecamp.org/news/how-i-established-a-good-release-process-in-javascript-b93e57e247e1/
https://docs.npmjs.com/cli/v8/commands/npm-version
https://git-scm.com/book/en/v2/Git-Basics-Tagging
https://github.com/devdigital/git-flow-standard-version
https://docs.npmjs.com/about-semantic-versioning
https://nodejs.dev/learn/semantic-versioning-using-npm
https://semver.org/spec/v2.0.0-rc.1.html
https://danielkummer.github.io/git-flow-cheatsheet/
http://git-scm.com/book/en/v2