Linee Guida

Jawa Druids

Versione	x.x.x
Data approvazione	xx-xx-xxxx
Responsabile	Nome Cognome
Redattori	Andrea Dorigo
Verificatori	Nome Cognome
Stato	Redazione in corso
Lista distribuzione	Jawa Druids Tullio NomeAziendaProponente
Uso	Interno

Sommario

Linee guide da seguire per una sessione lavorativa standard con git flow, Trello e Google Sheet.

Registro delle Modifiche

Modifica	Autore	Ruolo	Data	Versione
Riformulazione e formattazione	Andrea Dorigo	*Amministratore*	29-11-2020	v0.0.6
Aggiunta capitolo 3 e 4	Andrea Dorigo	*Amministratore*	29-11-2020	v0.0.5
Bugfix e formattazione	Andrea Dorigo	*Amministratore*	29-11-2020	v0.0.4
Riformulazione e formattazione	Andrea Dorigo	*Amministratore*	28-11-2020	v0.0.3
Revisione prima stesura	Margherita Mitillo	*Analista*	26-11-2020	v0.0.2
Prima stesura bozza	Andrea Dorigo	*Amministratore*	26-11-2020	v0.0.1

Indice

Introduzione

Lo scopo di questo documento è la stesura di un elenco di linee guida ed esempi che i componenti sono incoraggiati a seguire per migliorare l'efficacia della collaborazione. A differenza delle Norme di Progetto, questo documento è redatto in un linguaggio più informale per facilitarne la comprensione, con spezzati di codice a scopo esemplificativo e riferimenti esterni sulle best practices da seguire.
Il documento può essere soggetto a modifiche ed aggiunte per tutta la durata del progetto.

Git

2.1 Aggiunta o modifica di files

Poniamo che un componente del gruppo voglia aggiungere o modificare uno o più files nella repository del progetto. Il primo passo è creare una nuovo feature branch tramite il comando

```
git flow feature start analisi-uber
```

Questo crea un branch locale a partire dal develop chiamato `feature/analisi-uber` e vi esegue il checkout automaticamente; tutti i prossimi commit e push avverranno su questo branch, evitando l'immissione di errori o file ancora incompleti o non verificati sul `develop`. **Per rendere possibile la collaborazione sullo stesso branch** è possibile pubblicarlo in remoto con il comando

```
git flow feature publish analisi-uber
```

Chiunque desideri lavorare su questo branch necessita di immettere

```
 git checkout -b feature/analisi-uber origin/feature/analisi-uber
```

che crea un branch locale dallo stesso nome e lo sincronizza con quello remoto. Al completamento **e verifica** della nuova funzionalità, è possibile richiudere il branch nel `develop` ed eliminarlo tramite il comando

```
git flow feature finish analisi-uber
```

Verrà automaticamente generato un `merge commit`. Il responsabile di progetto si riserva la pubblicazione delle `release`.

2.2 Link utili e best practices sull'utilizzo di git

Potete trovare un elenco dei comandi di git flow più importanti e relative funzioni qui [1].
Un buon riassunto sull'utilizzo di git flow si può trovare a questo link [2].

Dell'altro materiale riguardante le best practice da seguire nei commit a questo indirizzo [3]. Fra le molte pratiche di cui tener conto nella scrittura del titolo del commit si evidenzia:

- esso deve contenere la descrizionee di cosa è stato fatto e perchè, ma non come.
- l'utilizzo dell'imperativo;

Per quest'ultimo punto vi riporto un paragrafo del link qui sopra che ne facilita la comprensione (è presente un errore nel testo dell'articolo, che ho corretto qui sotto):

```
Fixed the fencepost error //bad

Fixing the fencepost error //bad

Fix the fencepost error //good
```

The imperative mood is the one git commit message guideline that developers tend to violate most often. A good rule of thumb is that a git commit message can be appended to the statement "If applied, this commit will . . . ." The resulting sentence should make grammatical sense. As you can see from the following three examples, gerunds and past tense commits fail the test, while the imperative tense does not:

```
If applied, this commit will Fixed the fencepost error //bad

If applied, this commit will Fixing the fencepost error //bad

If applied, this commit will Fix the fencepost error //good
```

Trello

Trello è la piattaforma scelta dal gruppo per la gestione del progetto, data la famigliarità di certi componenti del gruppo con esso e la comprovata efficacia. Qui sono raccolti tutti gli stadi futuri/presenti/passati del progetto, e l'attività che ogni componente sta svolgendo.

La **creazione** e lo **spostamento** delle schede è riservata all'amministratore o al responsabile (o quantomeno è richiesta la loro approvazione). L'**assegnazione** delle schede è riservata al responsabile (o quantomeno è richiesta la sua approvazione).

Sessione lavorativa standard

Al termine di una sessione lavorativa, è bene:

1. eseguire l'aggiornamento del registro delle modifiche nella documentazione (se necessario)
2. committare e pushare, per evitare la perdita del lavoro effettuato
3. segnare le ore di lavoro svolte sulla Tabella delle ore lavorative [4].
4. aggiornare lo stato di avanzamento su Trello [5].

Bibliografia

[1] Daniel Kummer. *Git flow cheatsheet - efficient branching using git-flow by Vincent Driessen.* https://danielkummer.github.io/git-flow-cheatsheet/.

[2] Atlassian Bitbucket. *Git flow workflow tutorial.* https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow.

[3] Cameron McKenzie, TechTarget. *How to write a Git commit message properly with examples.* https://www.theserverside.com/video/Follow-these-git-commit-message-guidelines.

[4] Jawa Druids, 2020. *Tabella delle Ore Lavorative.* https://docs.google.com/spreadsheets/d/12esX1ISWQOKM-fjuHTLmAzRN0cWltksn7eGiPsBHBI0/edit?usp=sharing.

[5] Jawa Druids, 2020. *Trello - dashboard Jawa Druids.* https://trello.com/b/hIEOGbE9/jawadruids.