# Norme di progetto

Alt + F4

11 novembre 2024

# Registro Modifiche

| Versione | Data | Autore/i | Verificatore | Descrizione |
|---|---|---|---|---|
| v0.1 | 8 novembre 2024 | Pedro Leoni | Enrico Bianchi | Sottosezioni Guida a LaTeX, Creazione documento, Modifica documento, Verifica documento, Accettazione documenti, Versionamento, Tipi di documenti, Modalità di scrittura, Pubblicazione documentazione, Issue |

# Indice

# 1 Documentazione

## 1.1 Guida a LaTeX

### 1.1.1 Creazione tabella

```
\begin{table}[!h]
    \begin{tabularx}[| X | c | l | r |]
        \hline

        Intestazione1 &
        Intestazione2 &
        Intestazione3 &
        Intestazione4 \\

        \hline

        prima colonna prima riga   &
        seconda colonna prima riga &
        terza coolonna prima riga  &
        quarta colonna prima riga  &

        \hline

        ...

        \hline
    \end{tabularx}
    \caption{Riassunto contenuto}
\end{table}
```

Dove:

- `[!h]` indica che il posizionamento della tabella nel pdf deve essere dove si trova nel codice.
- `[| X | c | l | r |]` indica le colonne della tabella.

  `|` indica una riga di separazione tra le colonne.

  `X` indica una colonna che si allarga dinamicamente.

  `c`, `l` e `r` indicano rispettivamente colonne in cui il posizionamento del testo è centrale, a sinistra e a destra.
- `\hline` crea una linea orizzontale.

### 1.1.2 Link

Link ad elementi interni alla pagina:

```
\label[sec:nome]

...

\hypperref[sec:nome]{nomeLink}
```

Dove:

- Convezione standard per nominare le label: `tipoElemento:nome`.

  Esempi: img(immagini), sec(section), subsec(subsection), subsub(subsubsection), tab(tabella) ecc.

Link ad elementi esterni alla pagina:

```
\href{URL}{nomeLink}
```

### 1.1.3 Listati di codice

Listati di codice su più righe:

```
\begin{lstlisting}
    ...
\end{lstlisting}
```

Listati di codice inline:

```
\lstinline|code|
\lstinline+code+
```

### 1.1.4 Figure

```
\begin{figure}[h!]
    \includegraphics[scale=1.2]{pathImmagine}
    \caption{Riassunto immagine}
\end{figure}
```

Dove:

- `[h!]` permette di posizionare la figura nel pdf dove si trova nel codice.
- `scale=x.y` indica la scala da applicare all'immagine.

## 1.2 Creazione documento

Un documento viene creato solo se esiste una issue che lo prevede. La creazione di un documento segue i passi:

1. Presa in carico della issue all'interno della board-view del project GitHub. Questo richiede:
   (a) Auto assegnamento della issue.
   (b) Spostamento della issue nella colonna "In Progress" della board view dell'iterazione corrente.
2. Spostarsi sul branch develop tramite il comando:
   ```
   git checkout develop
   ```
3. Aggiornare il branch develop tramite il comando:
   ```
   git pull origin develop
   ```
4. Creazione e spostamento nel branch `feature/#id-issue` tramite il comando:
   ```
   git checkout -b feature/#id-issue
   ```
5. Creazione del documento usando template e nomi(esclusa versione) indicati nella sezione Tipi di documenti.
   Per i documenti di tipo: Verbali interni e Verbali esterni indicare la data in cui sono stati svolti mentre per gli altri documenti indicare la data attuale.
6. Popolazione del documento assicurandosi di soddisfare la definition of done indicata nella descrizione dell'issue.
   La scrittura deve rispettare la Modalità di scrittura.
7. Compilazione registro delle modifiche omettendo le colonne: Verificatore, Versione. Nella colonna Descrizione deve essere inserito un link agli elementi creati.
8. Notare che non viene modificato il nome del documento.
9. Registrazione delle modifiche nel repository locale tramite i comandi:
   ```
   git add *.tex
   git commit -m "messaggio-commit"
   ```
10. Aggiornare il ramo develop locale e mergiarlo nel ramo di feature tramite i comandi:

```
git checkout develop
git pull origin develop

git checkout feature/#id-feature
git merge develop
```

11. Pubblicazione del ramo nel repository tramite il comando:

    ```
    git push origin feature/#id-issue
    ```

12. Creazione di una pull request nel repository del progetto che parte dal ramo feature verso il ramo develop. La descrizione della pull request deve contenere il testo `closes id-issue`.

13. Aggiunta della pull request nella colonna "Backlog" della board view del progetto.

    `Progetto > (Backlog)+ Add item`

14. Assegnazione del valore "CurrentIteration" al field "Iteration" della pull request nella table view del progetto.

    `TabellaProgetto > Iteration > CurrentIteration`

## 1.3 Modifica documento

Un documento viene modificato solo se esiste una issue che lo prevede. La modifica di un documento segue i passi:

1. Presa in carico della issue all'interno della board view del project GitHub. Questo richiede:
   (a) Auto assegnamento della issue.
   (b) Spostamento della issue nella colonna "In Progress" della board view dell'iterazione corrente.

2. Spostarsi sul branch develop tramite il comando:

   ```
   git checkout develop
   ```

3. Aggiornare il branch develop tramite il comando:

   ```
   git pull origin develop
   ```

4. Creazione e spostamento branch `feature/#id-issue` tramite il comando:

   ```
   git checkout -b feature/#id-issue
   ```

5. Modifica del contenuto del documento assicurandosi di soddisfare la definition of done indicata nella descrizione dell'issue.

   La scrittura deve rispettare la Modalità di scrittura.

6. Compilazione registro delle modifiche omettendo le colonne: Verificatore, Versione. Nella colonna Descrizione deve essere inserito un link agli elementi modificati.

   Per i documenti di tipo diverso da Verbali interni e Verbali esterni deve essere aggiornata la variabile LaTeX chiamata `\date` usando la data attuale indicando il mese in lettere.

7. Notare che non viene modificato il nome del documento.

8. Pubblicazione del ramo nel repository tramite il comando:

   ```
   git push origin feature/#id-issue
   ```

9. Creazione di una pull request nel repository del progetto che parte dal ramo feature verso il ramo develop. La descrizione della pull request deve contenere il testo `closes #id-issue`.

10. Aggiunta della pull request nella colonna "Backlog" della board view del progetto.

    `Progetto > (Backlog)+ Add item`

11. Assegnazione del valore "CurrentIteration" al field "Iteration" della pull request nella table view del progetto.

    `TabellaProgetto > Iteration > CurrentIteration`

## 1.4 Verifica documento

Un documento deve essere verificato a seguito della sua modifica o della sua creazione. La verifica di un documento segue i passi:

1. Presa in carico della pull request all'interno della board view del project GitHub. Questo richiede:

   (a) Auto assegnamento della review della pull request.

   (b) Spostamento della pull request nella colonna "In Progress" della board view dell'iterazione corrente.

2. Ottenimento del ramo di feature tramite i comandi:

   ```
   git checkout -b feature/#id-issue
   git pull origin feature/#id-issue
   ```

L'id della issue relativa alla pull request si trova nella descrizione della pull request stessa.

3. Compilazione del documento e verifica del contenuto rispetto alla definition of done contenuta nella relativa issue.
4. Se la definition of done non è stata rispettata commentare la pull request indicando le parti mancanti.

   Il verificatore deve spuntare i task della definition of done che sono stati portati a termine.
5. Controllo grammaticale.
6. Aggiornare il registro delle modifiche popolando le colonne: Verificatore e Versione.

   Il cambiamento della versione richiede i seguenti passi:

   - Modifica della variabile LaTeX chiamata `\ultima-versione` seguendo la convenzione descritta nella sezione Versionamento. Questa variabile deve essere usata nella colonna Versione del registro delle modifiche.

     Occhio alla "v" che precede la versione.
   - Modifica del valore della colonna Versione nella riga precedente assegnando la versione indicata nel nome del documento sorgente.
7. Modifica del nome del documento assegnando la nuova versione.
8. Registrazione delle modifiche nel repository locale tramite i comandi:

   ```
   git add *.tex
   git commit -m "messaggio-commit"
   ```
9. Allineamento con lo stato attuale del ramo develop remoto tramite i comandi:

   ```
   git checkout develop
   git pull origin develop

   git checkout feature/#id-issue
   git merge develop
   ```
10. Pubblicazione del ramo nel repository tramite il comando:

    ```
    git push origin feature/#id-issue
    ```
11. Accettazione e merge della pull request.
12. Eliminazione del ramo di feature.

## 1.5 Accettazione documenti

L'accettazione dei documenti viene fatta quando è necessario pubblicare dei documenti compilati. Questo viene fatto quando è necessario mostrare all'esterno del team delle versioni "stabili" dei documenti. L'accettazione di un documento segue i passi:

1. Spostarsi sul branch develop tramite il comando:
   ```
   git checkout develop
   ```
2. Aggiornare il branch develop tramite il comando:
   ```
   git pull origin develop
   ```
3. Creazione e spostamento nel branch `release/NomeMilestone` tramite il comando:
   ```
   git checkout -b release/NomeMilestone
   ```
4. Controllo dei documenti da pubblicare.
5. Per ogni documento da pubblicare controllato:
   - Modificare il valore della variabile LaTeX chiamata `\ultima-versione` aumentando il numero di versione "stabile" e azzerando il secondo valore(spiegato nella sezione Versionamento).
   - Assegnare alla colonna dell'ultima riga del registro delle modifiche la versione indicata nel nome del documento.
   - Aggiungere una riga nel registro della versione in cui popolare tutte le colonne esclusa la colonna Verificatore.

     Nella colonna Versione usare la variabile LaTeX chiamata `\ultima-versione`. Nella colonna Descrizione usare il testo "Approvazione documento". Nella colonna data scrivere la data attuale indicando il mese in lettere.
6. Eventualmente apportare modifiche ai documenti e approvarli eseguendo i passi precedenti.
7. Eseguire un commit per registrare le modifiche nel repository locale tramite i comandi:
   ```
   git add *.tex
   git commit -m "messaggio-commit"
   ```
8. Merge del ramo di realese nei rami main e develop e pubblicazione delle modifiche in questi ultimi, tramite i comandi:

```
git checkout main
git merge release/NomeMilestone
git push origin main

git checkout develop
git merge release/NomeMilestone
git push origin develop
```

## 1.6 Versionamento

I documenti vengono versionati seguendo lo schema:

vX.Y

Dove:

1. X è un intero positivo non nullo aumentato a seguito della accettazione di un documento che si pensi resti invariato.

   L'aumento di questo valore genera automaticamente l'annullamento del secondo valore Y.

2. Y è un intero positivo non nullo aumentato al seguito della verifica di un documento.

## 1.7 Tipi di documenti

### 1.7.1 Verbali interni

1. Denominazione: `anno_mese_giorno-vX_Y.tex` dove anno, mese e giorno sono numerici e `vX_Y` è la versione del documento che segue lo schema di versionamento.
2. Template: utilizzare il template situato nel percorso `Template/verbale_interno.tex`.
3. Contenuto: Registro presenze, Verbale(Argomenti trattati e Decisioni prese) e To Do(associazione compiti-issue).
4. Cartella di destinazione: `Candidatura/VerbaliInterni`.

### 1.7.2 Verbali esterni

1. Denominazione: `nomeAzienda-anno_mese_giorno-vX_Y.tex` dove anno, mese e giorno sono numerici e `vX_Y` è la versione del documento che segue lo schema di versionamento.
2. Template: utilizzare il template situato nel percorso `Template/verbale_interno.tex`.

3. Contenuto: Registro presenze, Domande e Conclusioni.
4. Cartella di destinazione: `Candidatura/VerbaliEsterni`.

### 1.7.3 Norme Progetto

1. Denominazione: `normeProgetto-vX_Y.tex` dove `vX_Y` è la versione del documento che segue lo schema di versionamento.
2. Template: utilizzare il template situato nel percorso `Template/generico.tex`.
3. Contenuto: way of working del team.
4. Cartella di destinazione: `DocumentiInterni`.

## 1.8 Modalità di scrittura

- Preferire frasi brevi.
- Utilizzo di elenchi puntati.
- Usare sempre link a tabelle e figure.
- Indicare le caption delle tabelle e delle figure.
- Usare i comandi LaTeX relativi al codice per indicare: comandi, spezzoni di script, nomi file, percorsi e nomi tecnici.
- Scrivere i mesi delle date in lettere.
- Scrivere solo per risolvere un issue.
- Assicurarsi sempre di soddisfare la definition of done delle issue.

## 1.9 Pubblicazione documentazione

La pubblicazione della documentazione compilata avviene usando una GitHub Action. Questa GitHub Action compila i documenti sorgenti pubblicati nel ramo main del repository SorgentiDocumentazione e li pubblica nel ramo main del repository Documentaizone. Per fare ciò utilizza due variabili settate a livello di repository ovvero:

1. `DIRS_TO_DEL`: contiene un insieme di nomi di cartelle che devono essere eliminate a seguito della compilazione dei documenti. Se si vuole evitare di pubblicare una cartella il suo nome(preceduto da spazio) deve essere aggiunto al valore di questa variabile.

2. `DIRS_TO_IGNORE`: contiene un insieme di nomi di cartelle che devono essere lasciate invariate nel repository in cui vengono pubblicati i file compilati.

   Modificando il valore di questa variabile è quindi possibile aggiungere o togliere delle cartelle da non sovrascrivere.

   Queste cartelle attualmente sono solo quelle dei verbali esterni dato che vengono firmati dal referente dell'azienda.

## 1.10 Issue

Tutto ciò che viene fatto dal team deve essere documentato da issue di GitHub. Le issue per la documentazione devono avere una delle seguenti label:

1. bug: problematiche minori riguardanti il contenuto del documento associato all'issue. Es: errori ortografici e di formulazione delle frasi.
2. enhancement: automazioni che devono essere implementate.
3. documentation: creazione o modifica di un documento.
4. question: per approfondimenti sugli strumenti utili.

Il titolo di ogni issue deve essere un nome parlante che permette quindi d'identificare rapidamente il compito a essa associato.

Ogni issue deve contenere nella propria descrizione la definition of done sotto forma di check list. Questa permette al team di avere una definizione comune sullo stato in cui l'elemento riferito dalla issue soddisfa la necessità catturata dalla issue stessa. Una check list viene definita usando il seguente codice:

```
- [ ] primo elemento
- [ ] secondo elemento
...
```