Laurea Magistrale in informatica-Università di Salerno
Corso di *Gestione dei Progetti Software*-Prof.ssa F. Ferrucci e Prof. F. Palomba

Code Review

Riferimento	CR
Versione	2.0.0
Data	12/12/2024
Destinatario	Team di Code Review
Presentato da	Raffaella Spagnuolo, Alessia Ture
Approvato da	Raffaella Spagnuolo, Alessia Ture

Revision History

Data	Versione	Descrizione	Autori
23/11/24	1.0.0	Prima Stesura	AT
12/12/24	2.0.0	Revisione	RS

Project Managers

Nome	Acronimo	Contatto
Raffaella Spagnuolo	RS	r.spagnuolo6@studenti.unisa.it
Alessia Ture	ATu	a.ture@studenti.unisa.it

Sommario

1 Obiettivi della Code Review

1.1 Scopo del documento

Questo documento definisce le linee guida e il processo da seguire per effettuare una **code review tra pari** nel progetto *ZeroWaste Home*. L'obiettivo principale è garantire la qualità del codice, promuovere l'apprendimento reciproco e migliorare la collaborazione nel team.

1.2 Obiettivi

Gli obiettivi principali della code review includono:

- **Migliorare la qualità del codice**: Garantire che il codice sia leggibile, manutenibile, e conforme agli standard del progetto;
- **Individuare e correggere bug precocemente**: Ridurre i costi di correzione individuando errori prima del merge;
- **Condividere conoscenze**: Aiutare i membri del team a comprendere meglio le funzionalità sviluppate;
- **Standardizzare lo stile**: Assicurarsi che tutto il codice segua le stesse convenzioni e pratiche.

2 Strumenti Utilizzati

2.1 GitHub Pull Requests

Le pull request (PR) di **GitHub** rappresentano il fulcro del processo di code review. Attraverso le PR, i membri del team possono proporre modifiche al codice, ricevere feedback e collaborare per migliorare la qualità complessiva del progetto. Le funzionalità principali offerte da GitHub includono:

- **Tracciabilità delle modifiche**: Ogni PR documenta chiaramente i cambiamenti apportati e permette di associare le modifiche ai relativi task o requisiti;
- **Discussione centralizzata**: I revisori possono lasciare commenti direttamente sul codice, favorendo un confronto chiaro e costruttivo.

3 Processo di Code Review

3.1 Apertura di una Pull Request

Quando un membro del team completa una funzionalità o una modifica significativa al codice, deve:

- **Creare una PR** nel repository GitHub principale.
- Includere una **descrizione chiara e dettagliata** della PR:
  - Breve spiegazione del problema risolto o della funzionalità implementata,
  - Cambiamenti principali effettuati, con riferimenti a componenti, moduli o metodi,
  - Indicazione dei test effettuati e dei risultati ottenuti,
  - Eventuali problemi noti o punti da discutere.
- **Usare un template standard per la PR**:
  - **Titolo**: Una frase che descrive sinteticamente la modifica,
  - **Descrizione**: Strutturata con sezioni per componenti coinvolti, impatti sul sistema e obiettivi.

3.2 Assegnazione dei Revisori

Ogni PR deve essere assegnata a **tre revisori**. Tutti i revisori devono esaminare il codice con lo stesso livello di responsabilità, condividendo feedback su aspetti tecnici, leggibilità e conformità agli standard del progetto. L'obiettivo è ottenere una revisione **più dettagliata e completa**, sfruttando la prospettiva di ogni revisore.

Ogni revisore deve esaminare la PR nella sua totalità e fornire feedback su:

1. **Qualità del codice**: Verificare leggibilità, semplicità e modularità
2. **Aderenza ai requisiti**: Controllare che il codice soddisfi i requisiti funzionali descritti nella PR
3. **Documentazione**: Verificare che il codice sia commentato adeguatamente e che la PR contenga tutte le informazioni necessarie

Ogni revisore deve iniziare la revisione entro 24 ore dall'assegnazione. Tutti i revisori devono completare la loro revisione entro 48 ore.

3.3 Verifica del Codice

Durante la verifica del codice, i revisori devono analizzare ogni aspetto della PR seguendo criteri precisi. Gli elementi da verificare includono:

1. **Funzionalità**:

- La PR risolve il problema descritto o implementa correttamente la funzionalità richiesta?
- La logica è chiara e copre tutti i casi previsti?

2. **Stile e Convenzioni**:
   - Il codice rispetta le linee guida del progetto (es. Prettier)?
   - La struttura è coerente con il resto del progetto?
3. **Modularità e Manutenibilità**:
   - Le funzioni/metodi sono abbastanza modulari?
   - È evitata la duplicazione del codice?
4. **Test**:
   - Sono stati aggiunti test unitari adeguati?
   - I test coprono i casi principali e hanno successo?

I revisori devono annotare ogni problema o suggerimento nei commenti, specificando:

- **Cosa non va** (es. "La funzione è troppo lunga e difficile da leggere"),
- **Come migliorarlo**, se possibile (es. "Considera di suddividere questa funzione in metodi più piccoli").

3.4 Commenti e Feedback

I revisori possono discutere tra loro direttamente nei commenti della PR per chiarire eventuali dubbi o punti di disaccordo. In caso di divergenze significative, utilizzare il canale Slack per un confronto diretto. I revisori devono evitare di duplicare commenti simili. Se un problema è già stato segnalato, possono aggiungere un commento di supporto (es. "Concordo con quanto segnalato da X"). Tutti i commenti devono essere risolti dal creatore della PR prima dell'approvazione.

3.5 Risposta del Creatore della PR

Il creatore della PR ha il compito di affrontare i commenti e rispondere ai feedback ricevuti. Deve:

- **Rispondere a ogni commento**:
  - Fornire spiegazioni per le scelte tecniche, se necessario,
  - Indicare chiaramente come ogni problema è stato risolto.
- **Apportare le modifiche richieste**:
  - Aggiornare il codice in base ai feedback dei revisori,
  - Eseguire nuovamente i test per assicurarsi che tutte le modifiche siano funzionali.

- **Notificare i revisori**:
    - Informare i revisori una volta completate tutte le modifiche.

3.6 Approvo o Richiedo Modifiche

La PR può essere approvata solo se **tutti e tre i revisori concordano** che il codice è pronto per il merge. Se anche un solo revisore segnala problemi, la PR deve essere aggiornata e sottoposta a una nuova revisione. Le approvazioni devono essere fornite solo dopo che:

- Tutti i commenti sono stati risolti,
- I revisori sono soddisfatti delle modifiche apportate.

3.7 Merge della PR

Il merge della PR è la fase conclusiva del processo di code review. Deve avvenire solo dopo l'approvazione da parte di tutti i revisori. Il creatore della PR è responsabile di:

1. **Confermare l'approvazione**:
    - Verificare che tutti i commenti siano stati risolti,
    - Controllare che i tre revisori abbiano approvato formalmente la PR.
2. **Aggiornare il branch principale**:
    - Sincronizzare il branch della PR con il branch principale per risolvere eventuali conflitti,
    - Assicurarsi che il branch principale sia aggiornato con le modifiche più recenti.
3. **Eseguire il merge**:
    - Effettuare il merge nel branch principale tramite GitHub.
4. **Eliminare il branch**:
    - Eliminare il branch della feature una volta completato il merge, per mantenere il repository organizzata.

4 Best Practice per i Revisori

4.1 Prepararsi alla revisione

Prima di iniziare la revisione, è importante comprendere il contesto della PR. Leggere con attenzione la descrizione fornita dal creatore è il primo passo. Questo documento dovrebbe spiegare:

- Quale problema viene risolto o quale funzionalità viene aggiunta,
- Quali modifiche principali sono state apportate al codice,
- Quali test sono stati eseguiti e con quali risultati.

Avere una chiara comprensione degli obiettivi della PR consente al revisore di valutare il codice in modo mirato e approfondito. Inoltre, i revisori devono conoscere i requisiti funzionali e non funzionali relativi alla modifica, in modo da verificare che la PR soddisfi pienamente quanto richiesto.

4.2 Fornire Feedback Costruttivo

Il feedback fornito durante una revisione deve essere:

- **Chiaro e Specifico**: Ogni commento deve indicare esattamente cosa va migliorato e perché. Ad esempio, invece di scrivere "Questo metodo non mi piace", è meglio dire: "Questo metodo è troppo lungo e potrebbe essere suddiviso in più funzioni per migliorare la leggibilità";
- **Rispettoso e Costruttivo**: La revisione deve avere l'obiettivo di migliorare il codice, non di criticare il creatore. Usare un linguaggio positivo aiuta a mantenere un ambiente collaborativo;
- **Orientato alla Soluzione**: Se possibile, proporre suggerimenti pratici su come risolvere un problema.

I revisori devono anche collaborare tra loro per evitare di duplicare feedback già segnalati da altri. Se un problema è già stato evidenziato, è possibile aggiungere un commento di supporto (es. "Concordo con questa osservazione").

4.3 Linee guida per la verifica del codice

Tipo di codice	Domanda da Considerare	Esempio
Front-end(Angular)	Il componente fa quello che deve fare?	Il pulsante "Aggiungi Prodotto" aggiunge correttamente un prodotto?
	Il codice HTML è leggibile e i file CSS sono ben organizzati?	Gli elementi HTML hanno uno scopo chiaro?
	Sono gestiti gli errori dell'utente (es. campo vuoto)?	Mostra un messaggio se un campo obbligatorio non è compilato?
	Sono stati utilizzati correttamente i componenti di **PrimeNG**?	I pulsanti e i calendari sono creati con componenti PrimeNG invece di HTML puro?
	Le naming convention sono rispettate per componenti, variabili e file?	.
	Il codice è formattato correttamente con **Prettier**?	
Backend(Spring)	Le API funzionano come descritto nella PR?	Chiamando l'API /aggiungiProdotto, il prodotto viene aggiunto?
	Gli errori sono gestiti in modo chiaro e comprensibile?	Se manca un campo obbligatorio, l'API restituisce un messaggio utile?

	Il codice è facile da capire e segue uno stile coerente?	Ogni metodo ha un nome che spiega cosa fa (es. addProduct())?
	Checkstyle ha verificato il codice senza errori?	Non ci sono violazioni di stile secondo Checkstyle?
	Le naming convention sono rispettate?	
	Ogni classe e metodo contiene **JavaDoc** con descrizione e nome dell'autore?	
Testing	Sono presenti test che verificano il comportamento corretto?	
	Sono verificati anche i casi limite o situazioni in cui qualcosa va storto?	
	I test sono chiari e leggibili anche per chi non li ha scritti?	