Error_418

GitHub/Error-418-SWE error418swe@gmail.com

Verbale interno 06/03/24

Meeting post colloquio con Committente

Informazioni

Versione	1.0.1
Uso	Interno
Stato	Approvato
Responsabile	Carraro Riccardo
Redattore	Zaccone Rosario
Verificatore	Oseliero Antonio
Destinatari	Gruppo Error_418 Vardanega Tullio Cardin Riccardo

1 Informazioni generali

- Luogo: Discord$_G$
- Data e ora: 06/03/24 @ 09:00 ~ 10:00
- Partecipanti (6):
  - Carraro Riccardo
  - Gardin Giovanni
  - Nardo Silvio
  - Oseliero Antonio
  - Todesco Mattia
  - Zaccone Rosario
- Assenti (1):
  - Banzato Alessio

2 Ordine del giorno

A seguito dell'incontro con il Professor Cardin, il gruppo ha svolto un meeting interno riguardante:

- Considerazioni scaturite dal colloquio con il Professor Cardin;
- Pianificazione.

2.1 Considerazioni scaturite dal colloquio con il Professor Cardin

Il gruppo ha discusso i diversi argomenti trattati con il Professor Cardin durante il meeting.

2.1.1 Database$_G$

Il gruppo ha deciso di scegliere la linea di progettazione che vede il database$_G$ come parte del capitolato$_G$, e in particolare come data layer del software. Come suggerito dal Professor Cardin l'accesso avviene tramite backend anziché tramite browser$_G$ visti i potenziali problemi di sicurezza che questo comporterebbe.
La scelta è stata conseguita in seguito ai consigli del Professor Cardin, che ha anche portato l'esempio del pattern CQRS come caso di design dove un database$_G$ viene usato per sola lettura, come nel caso di WMS3. Il gruppo sceglie però di non implementare questo pattern, dato che nel suo schema è presente una parte dedicata alla scrittura dei dati che non è oggetto del capitolato$_G$.

2.1.2 Classi anemiche

Il gruppo ha deciso di proseguire con l'implementazione di classi che rappresentano le istanze del database$_G$. Il dubbio sull'implementarle o meno, poiché quest'ultime sono prive di comportamento specifico e risultano un semplice aggregato di attributi, è stato risolto grazie all'intervento del Professor Cardin, che nel meeting ha spiegato al gruppo il significato di questo tipo di classi, chiamate *anemiche*, e ha asserito che la loro implementazione è sensata, poiché il raggruppamento di dati correlati in una classe aiuta ad avere una struttura più pulita ed un'organizzazione più flessibile.

2.1.3 Business Logic

Il gruppo ha discusso sull'effettiva presenza della business logic in seguito al feedback ricevuto dal Professor Cardin. Questa è stata individuata principalmente nella parte che interroga l'API$_G$ REST

esterna e negli oggetti che rappresentano le istanze del database$_G$.
Si è discusso il modo in cui gli oggetti della business logic devono essere convertiti in oggetti della presentation logic, e si è posta la norma di non utilizzare gli oggetti ritornati dalle API$_G$ direttamente nella webapp.
In seguito alle osservazioni del Professor Cardin, si è deciso di implementare due diverse applicazioni, una per il backend e una per il frontend, entrambe con architettura a strati.

2.1.4 Design Pattern

Il gruppo, su consiglio del Professor Cardin, decide di inserire design pattern solo se ritenuti necessari, anche perché la loro implementazione porta ad overhead. I pattern ad ora individuati sono:
1) middleware: per separare frontend da backend;
2) provider: tipico in React$_G$ per passare variabili tra Componenti per evitare Prop Drilling.

2.1.5 Valutazione *Analisi dei Requisiti*$_G$

Il gruppo decide di proseguire con l'approfondimento dell'*Analisi dei Requisiti*$_G$, e si fissa come data massima di consegna per la rivalutazione la PB$_G$.

2.2 Pianificazione

ID task$_G$	Descrizione	Ruolo
WMS-80	Implementare la classe zona	Programmatore
WMS-81	Implementare la classe bin$_G$	Programmatore
WMS-82	Implementare la classe prodotto	Programmatore

Tabella 1: Task pianificate in seguito al colloquio con il Professor Cardin.