Progetto ingegneria del software

UNIMARKET

Davide Dell’Anno, Davide Carissoni, Francesca Corrente

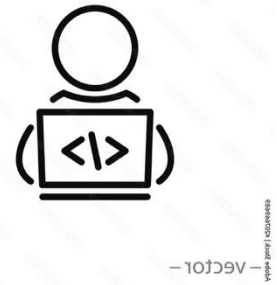
Obiettivo

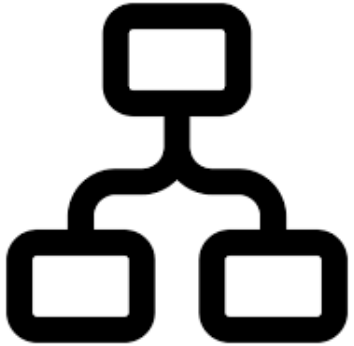
- UniMarket è stato progettato con l’obiettivo di semplificare la gestione della spesa per gli studenti.
- Si tratta di un’applicazione che consente di selezionare facilmente i prodotti desiderati e creare un carrello virtuale, ottimizzando il processo di ricerca e acquisto.

Difficoltà incontrate

- **JOOQ**:
  - integrazione
  - generazione tabelle
  - gestione
- **Vaadin**:
  - paywall per certe funzioni
  - implementazione funzionalità
- **Organizzazione**
- **Implementazione** funzionalità previste
- **Papyrus** e **creazione UML**

- **Programmazione:**
  - Programmazione ad oggetti
- **Linguaggi di programmazione**
  - Java
- **Modellazione** : Diagrammi UML
  - Use Case Diagram
  - Class Diagram
  - State Machinne
  - Sequence Diagram
  - Communication Diagram
  - Activity Diagram
  - Component Diagram
  - Package Diagram





Tools utilizzati

Eclipse IDE

Maven

SQLite

Vaadin

LaTeX

Papyrus

Software configuration management

Github

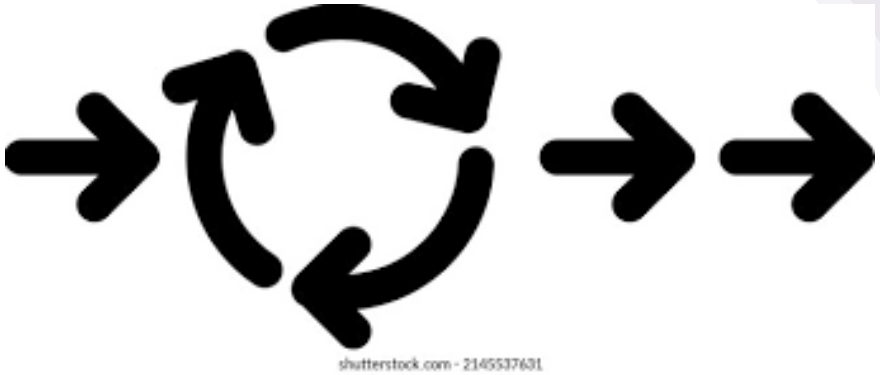
Issues, branches, pull request

Kanban Board

Software life cycle

RAD (Rapid Application Development**)**

Il modello di processo scelto per la realizzazione del progetto è basato su un approccio RAD (Rapid Application Development), integrato con tecniche Agile.



1. Analisi dei requisiti e prototipazione rapida

2. Sviluppo iterativo

3. Test continuo e revisione

4. Timebox

Requisiti

- Sono stati individuati immedesimandosi nell' Utente ed ispirandoci ad applicazioni simili già esistenti.
- Per garantire la qualità nello sviluppo del progetto, è stato adottato lo standard IEEE 830, che definisce linee guida per la stesura e la gestione delle specifiche dei requisiti software.
- Durante la fase di ingegneria dei requisiti l'obiettivo è stato quello di ottenere una descrizione chiara e completa dei problemi da risolvere.
- Dettagliatamente descritti nel documento denominato «Documentazione.pdf».

Architettura

Le viste architettoniche e i punti di vista utilizzati sono:

- **Modulo**: E` stato utilizzato il punto di vista a strati, che permette di visualizzare il sistema su livelli, i cui elementi possono utilizzare elementi di un livello inferiore.
- **Componenti e connettori:** E` stato utilizzato il punto di vista Client-Server, che descrive un sistema costituito da client e server cooperanti. In questo progetto, il client rappresenta l'interfaccia utente dell'applicazione e il server gestisce la logica di business e l'accesso ai dati.
- **Realizzativa:** E` stato utilizzato il punto di vista dell'implementazione, che de- scrive come i componenti logici e le funzionalità di alto livello vengono tradotti in strutture fisiche e tecniche.

Design pattern

I pattern utilizzati nel progetto sono:

- **Factory pattern:**

Implementato tramite il servizio Carrello Service, permette di garantire l'associazione di un carrello ad ogni utente al momento del login; se un utente ha già un carrello, questo viene recuperato dal database, altrimenti ne viene creato uno. Permette di evitare duplicazioni, migliora la manutenibilità e separa la logica dalla gestione.

- **Observer pattern:**

Implementato tramite due interfacce Observer: una per la gestione e una per l'interfaccia. L'oggetto osservato è il Carrello, tramite la gestione e notifica degli Observer in Carrello Service, permette di aggiornare in tempo reale le classi e interfacce che ne mostrano il contenuto.

Implementazione

- ✅ Registrazione
- ✅ Login
- ✅ Aggiunta al carrello
- ✅ Pagamenti
- ✅ Aggiunta prodotti

- ❌ Modifica disponibilità prodotti
- ❌ Logout
- ❌ Modificabilità account

Testing

- Manuale
  - Controlli sul corretto funzionamento tramite inserimento manuale dei dati
  - Controllo delle risposte del sistema all'inserimento di dati errati o nulli
- Automatico
  - Classi di test eseguite con JUnit
  - Controlli automatici per i metodi delle classi inserite
  - Problemi nel testare le interfacce con springboot

Demo

- Procediamo con una breve dimostrazione del funzionamento dell’applicazione creata.