Documentazione

UniMarket

Progetto di Ingegneria del Software

Di Davide Dell'Anno, Davide Carissoni, Francesca Corrente

Università degli Studi di Bergamo
Facoltà di Ingegneria Informatica
2024-2025

Indice

1 Project Plan

1.1 Introduzione

UniMarket è stato progettato con l'obiettivo di semplificare la gestione della spesa per gli studenti.
Si tratta di un'applicazione che consente agli studenti di selezionare facilmente i prodotti desiderati e creare un carrello virtuale, ottimizzando il processo di ricerca e acquisto.
L'applicazione non solo semplifica la gestione della spesa quotidiana, ma offre anche funzionalità pensate per le esigenze specifiche degli studenti, come la possibilità di salvare le proprie liste della spesa, monitorare i costi in tempo reale per rispettare il budget mensile e ricevere notifiche su offerte speciali o sconti.
UniMarket mira a migliorare l'esperienza di spesa online, offrendo una soluzione pratica ed efficiente per le esigenze quotidiane degli studenti.

1.2 Modelli di Processo

Lo sviluppo del progetto seguirà un processo **RAD** (Rapid Application Development), con l'obiettivo di realizzare un'applicazione funzionante entro i tempi stabiliti dal *time box*. Verrà adottato il **modello di Kano** per classificare i requisiti del progetto in base alla loro capacità di soddisfare le preferenze del cliente.

1.3 Organizzazione del Progetto

Per questo progetto è stata creata una **Squadra SWAT**, ovvero un team agile e versatile, composto da membri con competenze complementari, che si adatta rapidamente ai cambiamenti del progetto. In questo modo è più facile rispondere prontamente alle esigenze del cliente e dell'ambiente di sviluppo, mantenendo un'elevata efficienza nella gestione delle attività.
I ruoli sono stati assegnati come segue:

- **Team Leader: Davide Carissoni**
  Ha il compito di coordinare le attività, pianificare i moduli per gli sprint e assicurarsi il rispetto delle scadenze.
- **Sviluppatori, Progettisti, Tester: Francesca Corrente, Davied Carissoni, Davide Dell'Anno**
  Vista le piccole dimensioni del team, tutti i membri del progetto si occupano di tutte e tre le mansioni.

1.4 Standard, Linee Guida, Procedure

È molto importante che all'interno del progetto ogni componente del team coinvolto segua gli standard, le linee guida e le procedure concordate al fine di garantire un risultato facilmente comprensibile a tutti.
Gli standard Java Oracle definiscono le regole e le pratiche per la scrittura del codice, assicurando che il software sviluppato rispetti tali linee guida. In questo modo, l'approccio allo sviluppo adottato dal team garantirà uniformità e qualità.
Il linguaggio di programmazione scelto per lo sviluppo del software gestionale è Java.

Lo sviluppo avviene all'interno dell'**IDE Eclipse**, che permette la generazione automatica della documentazione tecnica tramite JavaDoc.
La Stesura della Documentazione del progetto è scritta utilizzando **LaTeX**, una scelta che semplifica la gestione della formattazione e il merge delle modifiche, garantendo un output professionale e ben strutturato. Per semplificare e velocizzare il processo di scrittura, è stato utilizzato il plugin LaTeX Workshop.
Per garantire una comprensione chiara e accurata del progetto, vengono utilizzati diagrammi **UML** (Unified Modelling Language) che permettono di visualizzare in modo intuitivo la struttura del sistema.
Per la gestione delle versioni e la condivisione del codice viene utilizzato **GitHub**, uno strumento essenziale per la collaborazione tra sviluppatori. GitHub consente di tracciare ogni modifica e mantenere il progetto sempre aggiornato. Inoltre, è stato utilizzato anche per la condivisione della documentazione e la gestione delle risorse correlate.
Per lo sviluppo dell'interfaccia grafica, verrà utilizzato **Vaadin**.

1.5 Attività di Gestione

Per garantire una gestione efficace delle attività e dello sviluppo del progetto, sono stati definiti i seguenti obiettivi:

- **Incontri settimanali:** monitorare l'avanzamento del lavoro, affrontare eventuali problemi riscontrati, discutere strategie risolutive e valutare la direzione dello sviluppo.
- **Kanban board su GitHub:** tracciare il progresso delle attività, organizzare il lavoro e assegnare incarichi ai membri del team.
- **Gestione del progetto con GitHub:** tenere traccia delle modifiche, monitorare gli aggiornamenti e facilitare la collaborazione.

1.6 Rischi

I rischi a cui bisogna prestare più attenzione sono:

- **Compatibilità dei sistemi utilizzati:** Il progetto viene sviluppato per essere utilizzato su desktop; pur usando maven, che consente il download automatico delle librerie richieste, potrebbero comunque esserci problemi di visualizzazione o calcolo su mobile o altri S.O. desktop.
- **Limitate quantità e tipologie di prodotto:** UniMarket è indipendente da reali supermarket, non dispone di tante opzioni per ogni genere di prodotto.
- **Bugs:** Ci potrebbero essere problemi generati dall'uso scorretto e imprevisto del programma da parte del cliente, è possibile che alcuni bug agli estremi dell'utilizzo improprio rimangano pur dopo i test.

1.7 Personale

Il team è composto da tre membri:

- **Francesca Corrente**

- **Davide Carissoni**
- **Davide Dell'anno**

Ogni membro avrà funzioni simili all'interno del team, con ruoli assegnati in base alle esigenze, senza l'intervento di personale esterno.

1.8 Metodi e Tecniche

I metodi e le tecniche adottati per le diverse fasi dello sviluppo sono i seguenti:

- **Pianificazione dei requisiti:** si basa sull'elicitazione e sull'analisi dei requisiti per definire in modo chiaro le funzionalità richieste dall'applicazione.
- **Progettazione dell'applicazione:** prevede la definizione del funzionamento dell'applicazione, delle funzionalità incluse e degli strumenti software da utilizzare.
- **Programmazione:** consiste nello sviluppo del codice sorgente dell'applicazione seguendo le specifiche progettuali.
- **Testing:** include la scrittura e l'esecuzione di test, utilizzando strumenti appositi, per verificare il corretto funzionamento del codice e individuare eventuali anomalie.

Queste attività saranno supportate da un'adeguata gestione del controllo di versione e della configurazione dei componenti software, documentando ogni fase con precisione.

1.9 Garanzie di Qualità

Al fine di assicurare la qualità durante il processo di sviluppo e la realizzazione del software, verranno utilizzati i seguenti criteri:

- **Correttezza:** Il software deve funzionare come previsto e soddisfare i requisiti specificati, senza errori o comportamenti imprevisti. La correttezza implica che il sistema esegua tutte le operazioni correttamente, restituendo i risultati attesi in ogni condizione di input.
- **Affidabilità:** Il software deve essere stabile e operare in modo continuo e senza interruzioni. Un software affidabile deve resistere a guasti o malfunzionamenti, gestendo situazioni impreviste e mantenendo la sua funzionalità nel tempo, anche sotto carichi elevati o condizioni di stress.
- **Integrità:** L'integrità riguarda la protezione dei dati e il mantenimento della coerenza e correttezza durante le operazioni. Un software con buona integrità assicura che i dati non vengano alterati o danneggiati involontariamente, né durante l'elaborazione né durante la memorizzazione.
- **Usabilità:** Il software deve essere intuitivo e facile da usare per gli utenti finali. Una buona usabilità garantisce che gli utenti possano navigare nell'interfaccia senza difficoltà, riducendo il numero di errori operativi e migliorando l'esperienza complessiva.
- **Manutenibilità:** Un software manutenibile è facilmente aggiornabile e modificabile nel tempo per adattarsi a nuove esigenze o correggere eventuali difetti. Ciò implica che il codice sia chiaro, ben strutturato e documentato, facilitando interventi futuri senza introdurre regressioni o problemi aggiuntivi.

1.10 Paccheti di lavoro

Nel team, il lavoro sarà suddiviso e gestito attraverso l'uso della Kanban Board, uno strumento che offre una visione chiara e immediata del flusso di lavoro.
La board consente di monitorare le milestone, visualizzare l'avanzamento delle attività che vengono completate o che sono in fase di sviluppo, e assegnare tasks.

1.11 Risorse

Le risorse relative allo sviluppo software sono le seguenti:

- **Eclipse IDE:** per lo sviluppo e la modifica del codice Java.
- **Maven:** per la gestione e l'automazione dei progetti Java.
- **SQLite:** per la creazione e gestione di un database embedded.
- **Vaadin:** per la progettazione iniziale dell'interfaccia grafica dell'applicazione.
- **JUnit:** per la sccrittura ed esecuzione di test.

Strumenti aggiuntivi:

- **LaTeX:** per la redazione della documentazione tecnica.
- **Papyrus:** per la modellazione di diagrammi UML.

Le risorse hardware si limitano ai dispositivi personali dei membri del team.

1.12 Budget

Il budget del progetto, essendo privo di spese finanziarie, è relativo al tempo di lavoro necessario, ipotizzato intorno alle 60 ore per ciascun membro.

1.13 Cambiamenti

Le modifiche principali saranno apportate durante la fase di ingegneria dei requisiti, per garantire un prodotto il più possibile aderente alle richieste. Eventuali modifiche al codice, derivanti dalla scrittura di test o da cambiamenti delle funzionalità dell'applicazione, verranno gestite attraverso la creazione di un nuovo branch nel repository, accompagnata dall'aggiornamento della documentazione correlata.

1.14 Consegna

La documentazione e l'applicazione finale saranno consegnate condividendo il progetto sul repository GitHub.

2 Gestione del progetto

2.1 Software Life Cycle

Il modello di processo scelto per la realizzazione del progetto è basato su un approccio **RAD** (Rapid Application Development), integrato con tecniche **Agile**, per garantire velocità, flessibilità e un'elevata capacità di adattamento alle esigenze degli utenti.
L'obiettivo di questo tipo di processo è fare uso delle tecniche **Agile** per bilanciare la mancanza di esperienza dei membri del team, dando priorità alla comunicazione e collaborazione, al fine di garantire velocità di realizzazione, qualità del sistema, flessibilità, iterazione.

2.2 Processo

Il processo è suddiviso in diverse fasi, ciascuna caratterizzata da attività iterative e collaborazione costante tra i membri del team:

1. **Analisi dei requisiti e prototipazione rapida:** Inizialmente vengono raccolti i requisiti fondamentali per comprendere appieno le esigenze degli utenti e le aspettative del cliente. Successivamente, vengono creati prototipi preliminari che permettono di raccogliere feedback, consentendo così di perfezionare e verificare le funzionalità sin dalle fasi iniziali del progetto.

2. **Sviluppo iterativo:** Ogni iterazione comprende le fasi di pianificazione, progettazione, implementazione e test. Le funzionalità vengono sviluppate in blocchi incrementali, consentendo il rilascio continuo di versioni con qualità elevata.

3. **Test continuo e revisione:** I test vengono eseguiti frequentemente durante tutto il processo per garantire un elevato standard qualitativo e per identificare tempestivamente eventuali problematiche. Il feedback continuo è fondamentale nell'orientare lo sviluppo del software. Questo processo permette di identificare le aree di miglioramento e di adattare il software a nuove esigenze in modo rapido e mirato.

4. **Timebox:** Il team ha suddiviso il lavoro in *timebox* della durata di una o due settimane, durante le quali ogni membro si è focalizzato su uno specifico obiettivo, contribuendo in modo mirato e collaborativo all'avanzamento del progetto. Essendo il gruppo composto da tre membri, la comunicazione dei processi e le eventuali revisioni del progetto sono state tempestive e su base giornaliera.

   Calendario timebox alla fine del documento.

2.3 Gestione della configurazione

Per la gestione della configurazione è stato adottato **GitHub**, utilizzato in combinazione con il tool **GitHub Desktop** per il controllo dei repository locali e le operazioni di *pull*, *commit* e *push* eseguite dai membri del team.
La distribuzione del lavoro è organizzata tramite una **Kanban board** integrata su GitHub, con cui, attraverso la creazione di *issues*, quando necessario, vengono definiti i compiti da svolgere e assegnati ai relativi membri del team, al fine di sistematizzare e ottimizzare il processo di sviluppo. L'uso di **milestones** permette di monitorare lo stato di avanzamento degli *issues* e del completamento delle *task*.

Durante l'implementazione, eventuali errori e problematiche sono notificati e vengono tempestivamente segnalati e gestiti tramite l'uso di *issues*. Ogni *issue* viene aperta dal membro del team che individua il problema e successivamente assegnata a uno o più colleghi competenti, che si occuperanno di risolverla.
La gestione degli *issues* e l'utilizzo di **branch** dedicati per le nuove versioni del prodotto hanno l'obiettivo di assicurare un approccio strutturato e accurato durante tutte le fasi dello sviluppo del progetto. L'implementazione e la revisione del codice saranno effettuate mediante la creazione di *pull request* da parte dei membri del team, prima di procedere al *merge* sul *branch* di destinazione.
La repository del progetto è stata organizzata nel seguente modo:

- **Documentazione:**
- **Media:**
- ...

2.4 People Management

Il team presenta caratteristiche simili a quelle di un'unità **SWAT**, ma il livello di competenze tecniche dei membri non è particolarmente avanzato. Di conseguenza, vengono adottati principi propri di un **team Agile**, con particolare attenzione alla divisione del lavoro e alla comunicazione tra i membri.
Tutti i membri possiedono competenze tecniche simili, pertanto i ruoli non vengono assegnati in base a specifiche abilità, ma ogni membro partecipa attivamente a tutte le fasi dello sviluppo.
La suddivisione delle attività è decisa collettivamente durante i *timebox*, con la possibilità di ridefinire i ruoli, se necessario.
Non viene istituita una gerarchia formale all'interno del gruppo, sebbene sia presente un **leader** il cui unico compito è garantire la correttezza e la puntualità del lavoro svolto.

3 Requisiti

4 Design

5 Testing

6 Maintenance

Calendario Timebox

SETTIMANA/E	OBIETTIVO	INCONTRI
4/11/2024 - 17/11/2024	Project Plan	6
18/11/2024 - 24/11/2024	Documentazione, gestione del progetto	3
25/11/2024 - X/12/2024	Documentazione, Specifica dei requisiti + UML	