*Project Plan di sviluppo del software: QuickByte, un’applicazione per la gestione di ordini di cibo a domicilio, che collega: ristoranti, clienti e corrieri attraverso un'interfaccia intuitiva.*

PROJECT PLAN:

**1. Introduzione**

Durante l’anno 2024, un team di tre studenti di Ingegneria Informatica decide di sviluppare un’applicazione per la gestione di un sistema software di delivery food, data la scarsa presenza di prodotti simili e funzionali sul mercato.

L’obiettivo del progetto è di sviluppare una piattaforma che permetta all’utente, che può essere di tipo cliente, titolare di un ristorante o corriere, di: effettuare ordini, ampliare le modalità di vendita o guadagnare completando consegne.

La squadra è composto da: Anes Hamza, Bonomelli Pietro e Rota Leonardo

-**TODO**: conclusioni-

**2. Modello di processo**

Il modello del ciclo di vita scelto è stato il Rational Unified Process (RUP), un modello iterativo che può essere considerato come una via di mezzo tra i metodi basati su documenti e agili. Non è stato scelto un modello puramente agile per una necessità di avere una documentazione corretta e completa.

Dividiamo quindi il ciclo di vita del software nelle 4 fasi: (fonte wikipedia)

*Fase iniziale*, si concentra sull’ottenimento di obiettivi chiari. Questo include la pianificazione iniziale del progetto, la valutazione dei rischi, una definizione grossolana dei requisiti e così via. Se il progetto non supera questa milestone, detta "Lifecycle Objective Milestone", esso dovrà essere abbandonato o ridefinito.

*Fase di elaborazione*, nella quale verranno portati a termine questi punti fondamentali:

deve essere stato sviluppato un modello dei casi d'uso completo all'80%

dev'essere fornita la descrizione dell'architettura del sistema

dev'essere stata sviluppata un'"architettura eseguibile" che dimostri il completamento degli use case significativi

dev'essere eseguita una revisione del business case e dei rischi

dev'essere completata una pianificazione del progetto complessivo

*Fase di costruzione,* in questa fase viene portato a termine il grosso degli sviluppi. Viene prodotta la prima release del sistema. La milestone di questa fase si chiama "Initial Operational Capability" e rappresenta la prima disponibilità delle funzionalità del sistema in termini di implementazione. DEFINIRE MILESTONE

*Fase di transizione*, il sistema passa dall'ambiente dello sviluppo a quello del cliente finale. Vengono condotte le attività di training degli utenti e il beta testing del sistema a scopo di verifica e validazione. Si deve in particolare verificare che il prodotto sia conforme alle aspettative descritte nella fase di Inception. Se questo non è vero si procede a ripetere l'intero ciclo; altrimenti, si raggiunge la milestone detta "Product Release" e lo sviluppo termina.

Oltre al modello RUP, seguiremo anche un approccio Model Driven Architecture (MDA), in particolare progetteremo diagrammi UML tramite il plugin Papyrus installato sulla macchina virtuale Eclipse, ambiente di sviluppo che utilizziamo per scrivere codice Java.3. Organizzazione del progetto

**3. Organizzazione del progetto:**

Il team di sviluppo si interfaccerà con i professori che seguono lo sviluppo dell’applicazione.

Non essendoci utenti, non ci sarà interazione con degli utenti. Noi svilupperemo il software che, dopo la prima fase di implementazione, verrà presentata una versione di prova e testeremo noi stessi se stiamo seguendo le linee guida e gli obiettivi che c’eravamo posti, quindi una verifica e validazione dopo ogni fase.

**Ruoli:**

essendo in tre sviluppatori i ruoli saranno di tipologia Agile: non esiste una gerarchia specifica, ergo è necessaria un’adeguata autodisciplina ed un’ottima capacità di adattamento.

Tutti gli esperti del team devono essere versatili in quanto dovranno svolgere le diverse funzioni necessarie (progettazione, sviluppo, testing, ecc.).

La comunicazione sarà tempestiva e poco formale tra i membri del team e la documentazione moderatamente formale.

Conoscenze richieste minime: programmazione Java, uso della lingua italiana.

**4. Standard, linee guida, procedure**

-TODO

**5. Attività di gestione**

Organizzeremo brevi riunioni con cadenza aperiodica per analizzare lo stato di avanzamento del progetto, producendo un breve report come output.

Questa fase sarà gestita anche attraverso GitHub, monitorando lo sviluppo delle classi Java.

**6. Rischi**

1. Mancanza di un membro del team a causa di altri impegni (es. corsi universitari).  
   **Soluzione:** Adottare un'organizzazione del lavoro flessibile e documentare tutte le decisioni prese durante le riunioni. Inoltre, assegnare backup temporanei per i compiti critici.
2. Possibili difficoltà con tecnologie e framework nuovi. **Soluzione:** Investire nell'autoformazione, consultare documentazione ufficiale e risorse online, ed eventualmente suddividere il lavoro per sfruttare l'expertise di ciascun membro.
3. Comportamento diverso del sito web su vari browser  
   **Soluzione:** Testare il codice su una varietà di browser e sistemi operativi fin dalle prime fasi di sviluppo per identificare problemi di compatibilità.
4. Ritardi nella consegna delle milestone  
   **Soluzione:** Adottare una gestione Agile, monitorando costantemente lo stato di avanzamento del progetto.
5. Vulnerabilità nel sistema di autenticazione o nelle transazioni di pagamento.  
   **Soluzione:** Implementare buone pratiche di sicurezza, come la crittografia dei dati.
6. Comunicazione inefficace all’interno del team **Soluzione:** Utilizzare strumenti di gestione del progetto e pianificare meeting settimanali per aggiornamenti.
7. Bug non rilevati durante la fase di test  
   **Soluzione:** Ampliare i casi di test, coinvolgere beta tester esterni.
8. Scarsa accettazione da parte degli utenti finali  
   **Problema:** L’app potrebbe non soddisfare le aspettative dei clienti, ristoratori o corrieri, in quanto potrebbe non risultare chiara e/o intuitiva.  
   **Soluzione:** Coinvolgere più utenti e beta tester, affinché aiutino a mettere luce sui problemi dell’applicazione, nelle fasi di prototipazione e raccolta di feedback continuo.

**7. Personale**

Nel nostro team non esiste una gerarchia specifica, quindi tutti e tre gli sviluppatori copriranno tutti i ruoli durante le varie fasi del progetto. Vedi i punti 6.1 e 6.6 per eventuali rischi.

**8. Metodi e tecniche**

OS utilizzati per lo sviluppo:

* Windows 10
* Windows 11
* Ubuntu 24.04

Test (capitolo 13)

**9. Garanzia di qualità**

*(Quale organizzazione e procedure verranno utilizzate per garantire che il software in fase di sviluppo soddisfi i requisiti di qualità dichiarati?*

*I molti aspetti di un Piano di Assicurazione della Qualità possono anche essere trattati in un documento separato. Il tema della garanzia della qualità è discusso nel Capitolo 6 .)*

Adotteremo le Best Practices nel coding:

* Variabili globali definite in maiuscolo
* Nome dei metodi e delle variabili locali in notazione camelCase
* Indentazione standard
* ecc…

**10. Pacchetti di lavoro (workpackages)**

**Documentazione**

Redazione del Project Plan (vedi punto 14)

Aggiornamento continuo per allineare il progetto con le modifiche approvate.

**Progettazione**

Progettazione dell’architettura software (definizione delle componenti principali e delle loro interazioni).

Progettazione del database (schema ER, file csv embedded).

**Costruzione del database embedded  
 Programmazione Backend**

Gestione dell’autenticazione e dei ruoli (clienti, ristoranti, corrieri, utenti).

Implementazione delle funzionalità principali, come: Registrazione utenti, gestione ordini, integrazioni con il database.

**Programmazione GUI**

Sviluppo dell’interfaccia utente per gli utenti che ne faranno uso.

Realizzazione di pagine intuitive.

**Testing**

Test unitari: Verifica delle singole componenti (funzioni, moduli, ecc.).

Test d’integrazione: Controllo del corretto funzionamento tra frontend, backend e i file che compongono il database.

Test di sistema: Valutazione del comportamento dell’app nell’ambiente finale.

Test di accettazione: Coinvolgimento del professore e beta tester per verificare il rispetto dei requisiti.

**11. Risorse**

Risorse hardware:

* PC
* Spazio condiviso Google Drive
* Connessione wi-fi

Risorse software:

* Eclipse versione 2024-09
* Maven
* Papyrus
* Discord, Google Meet, applicazioni di messaggistica per la comunicazione in tempo reale
* Github (desktop)

**12. Budget e programma**

Nello sviluppo del software non è previsto un budget economico specifico, ma è importante avere una stima dei tempi in modo accurato. Questo perché il progetto ha una scadenza fissata per gennaio 2025, con l’attività iniziata a novembre 2024.

La stima dei tempi sarà quindi uno strumento essenziale per garantire il rispetto della deadline, suddividendo il lavoro in fasi ben definite e monitorando costantemente lo stato di avanzamento.

**13**. **Cambiamenti**

**Registrazione delle modifiche proposte:** Ogni modifica deve essere documentata in un registro centralizzato, specificando la natura del cambiamento, la ragione della proposta e l'urgenza.

**Revisione e approvazione:** Le modifiche proposte saranno discusse durante le riunioni del team, e approvate solo dopo aver valutato l’impatto sul progetto.

**Stima dell’impatto:** Una volta approvata, la modifica sarà analizzata in termini di tempi di implementazione e impatti sulle funzionalità già esistenti, e quindi sulle qualità del software.

**Implementazione controllata:** La modifica verrà incorporata nel progetto seguendo le procedure definite, assicurandosi che sia accompagnata da:  
 - Aggiornamenti al codice con commenti chiari.  
 - Modifica della documentazione correlata.  
 - Test per garantire che non vengano introdotti bug.

**Versionamento e controllo della configurazione:** Per gestire le diverse versioni di codice e documentazione si userà GitHub, che consentirà di tenere traccia dei cambiamenti e di ritornare a una versione stabile in caso di problemi.

**14. Consegna**

Il Project Plan, che include la pianificazione dettagliata, la gestione dei rischi e dei cambiamenti, deve essere consegnato al principale stakeholder, il professor Gargantini, almeno un mese prima della consegna finale del progetto.

Prima della consegna definitiva, il sistema sarà sottoposto a:  
***Test di accettazione*** (User Acceptance Testing) per verificare se sono soddisfatti i requisiti,  
**Unit test**

Verrà consegnata al cliente una **documentazione completa e accurata**, includendo: Manuale utente e il Registro delle modifiche effettuate durante lo sviluppo.

I documenti consegnati devono contenere:

* Project Plan,
* SW life cycle,
* Configuration management,
* People Management and Team Organization,
* Software Quality,
* Documento sui requisiti,
* UML,
* Descrizione sull’architettura,
* Design del SW,
* Test effettuati sul SW,
* Refactoring/documentazione sulla manutenzione del sw.