**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)

1. 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.
2. 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

1. 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.
2. 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**

**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**

Potenziali rischi che potremmo incontrare:

* Mancanza di un membro del team a causa di altri impegni (come corsi universitari). Per questo motivo, adotteremo un'organizzazione del lavoro flessibile e documenteremo tutte le decisioni prese durante le riunioni.
* Possibili difficoltà con tecnologie e framework nuovi. Per affrontarle, ci impegneremo nell'autoformazione e nella consultazione di documentazione e risorse online.
* Il sito web potrebbe comportarsi in modo diverso su diversi browser. Una possibile soluzione è testare il codice su vari browser.

**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.

**8. Metodi e tecniche**

**9. Garanzia di qualità**

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

**11. Risorse**

**12. Budget e programma**

**13. Cambiamenti**

**14. Consegna**