A picture containing logo

Description automatically generated

DOCUMENTAZIONE

Componenti del gruppo:

Carminati Matteo n°matricola: 1066354

Torri Lorenzo n°matricola: 1069047

Università degli Studi di Bergamo - Corso di Ingegneria del Software – a.a. 2021/2022

**CAPITOLO 3 – Software Life Cycle**

La scelta del modello di vita del nostro software è ricaduta su uno di tipo agile, in particolare XP. Questa scelta è legata all’organizzazione del team che è composto esclusivamente da due persone, inoltre c’è la necessità di apportare al progetto rapidi e funzionali cambiamenti.

Extreme Programming è ottimale per il team in quanto la gestione del lavoro si basa interamente sul pair programming. I due componenti del team infatti lavorano per la maggior parte del tempo insieme riuniti in calls di Microsoft Teams. Entrambi conoscono perfettamente ogni aspetto del progetto sviluppato. Ogni componente del team è in grado di cambiare il codice, in quanto lo conosce e lo sa manipolare correttamente.

Per lo sviluppo del codice i due sviluppatori hanno deciso di dividersi il lavoro, tuttavia le modifiche saranno sempre sottoposte a verifica da parte dell’altro membro. Tali verifiche saranno effettuate durante brevi riunioni da remoto in modo che si possa avere un confronto diretto e immediato.

L’idea è quella di iniziare a lavorare su una prima versione basilare ma funzionante dell’applicazione, per poi rilasciare costantemente delle piccole versioni del sistema che vadano a migliorare la situazione di partenza e che permettano di verificare le funzionalità implementate.

Al termine dell’introduzione di una nuova funzionalità all’interno della applicazione bisognerà fare alcuni test. La fase di testing ha un ruolo chiave e verrà affrontata con più precisione in seguito (capitolo 13).

Il team non si focalizzerà sulla parte grafica, terrà infatti un design grafico che sia il più semplice possibile, proprio perché quella commissionata è solo un prototipo della vera applicazione che potrà essere implementata in futuro.

Come ultima precisazione il team predilige, come già specificato nel Project Plan al punto 8, un approccio di tipo Model driven architecture, dunque, quando possibile, i cambiamenti dell’applicativo vengono effettuati sui modelli UML implementati e non direttamente sul codice.

La software life cycle è stata parzialmente implementata da parte degli sviluppatori anche mediante formalizzazione eseguita con il diagramma statechart del tool UML.

Tale diagramma è reperibile all’interno del file “Diagrammi UML.mdj” inserito nella repository GitHub sotto la voce ”Documentazione UML” , “CAPITOLO 3” con il nome di “SW\_Life\_Cycle”.

**CAPITOLO 5 – People Management and team Organization**

L'organizzazione del team e la gestione delle persone che ne fanno parte è riconducibile perlopiù all'approccio agile, i due sviluppatori infatti collaborano in ogni aspetto del loro lavoro, applicando anche pair programming.

Per certi aspetti la gestione dei membri del team viene affrontata seguendo lo schema proposto da Mintzberg; in particolare vi sono alcune analogie con il meccanismo di coordinazione di adhocracy. L’organizzazione del lavoro avviene infatti prediligendo meeting e momenti di scambio di idee tra i due membri del team.

Altro aspetto tipico di adhocracy, e ripreso dal team di sviluppo, è la scelta di coordinare il lavoro attraverso mutual adjustment che i membri svolgono con molta regolarità.

Tali correzioni avvengono però tenendo sempre presente quelli che sono gli obiettivi prefissati, ponendo il focus sul prodotto finale considerato nella sua completezza e mai suddiviso in parti.

Per quanto riguarda la strutturazione del team di lavoro, questa si discosta da adhocracy e si addice molto di più ad un modello agile; all’interno del team infatti non esiste alcun tipo di gerarchia o di rigida suddivisione dei compiti.

Questo approccio è stato ritenuto quello più adatto poiché i due componenti del team hanno percorsi formativi e skills molto simili, pertanto svolgono compiti analoghi e godono entrambi di grande autonomia non dovendo rispondere alle direttive di un superiore.

Lo svolgimento del lavoro segue sempre gli standard legati ai linguaggi utilizzati.

Il team ha solo specifiche indicative riguardanti l’ammontare di ore di lavoro richieste per lo sviluppo della piattaforma.

Per poter organizzare meglio il workflow gli sviluppato hanno deciso di utilizzare il servizio Kanban integrato a Github che permette, con le sue boards, di visualizzare e scegliere gli obiettivi di giornata.

**CAPITOLO 6 - Software Quality**

Il cliente non ha specificato dei requisiti di qualità relativi alla piattaforma richiesta, pertanto questi sono stati determinati direttamente dal team di lavoro.

In particolare i due sviluppatori hanno deciso di seguire le direttive del modello McCall per avere dei criteri qualitativi da rispettare durante il lavoro svolto.

Alcuni parametri qualitativi saranno valutati in fase di sviluppo, altri necessariamente verranno presi in considerazione solo al termine dell’applicativo, o comunque durante le fasi finali di sviluppo.

Le categorie dei fattori di qualità del software individuati dal modello McCall sono:

* Product operation
  + Correctness
  + Reliability
  + Efficiency
  + Integrity
  + Usability
* Product revision
  + Maintainability
  + Testability
  + Flexibility
* Product transition
  + Portability
  + Reusability
  + Interoperability

**CAPITOLO 9 – Requirements Engineering**

Come spiegato già all’interno del project plan l’applicativo sviluppato è al momento solo un prototipo di quella che potrebbe essere la piattaforma di prenotazione online definitiva e commercializzabile.

Il lavoro non è stato commissionato dal Presidente dell’associazione, ma è una proposta che i due sviluppatori hanno deciso di presentare. Pertanto non sono stati forniti dal Presidente requisiti particolari inerenti alla piattaforma stessa. Il team tuttavia, basandosi anche sull’esperienza dei singoli membri nell’ambito delle prenotazioni di strutture sportive, ha provveduto a stilare una serie di requisiti che vorrebbero vedere implementati in una piattaforma di questo genere.

I requisiti nella fase di elicitation sono stati organizzati in ordine gerarchico seguendo il modello Moscow:

* Must have
  + L’utente ha la possibilità di scegliere lo sport che vuole praticare.
  + L’utente ha la possibilità di scegliere l’orario di prenotazione.
  + L’utente ha la possibilità di scegliere la struttura che vuole prenotare.
  + La piattaforma cerca strutture con le indicazioni dell’utente.
  + L’utente conferma la prenotazione.
  + La piattaforma calcola il costo della prenotazione.
* Should have
  + L’utente ha la possibilità di scegliere se desidera avere lo spogliatoio o no.
  + L’utente ha la possibilità di effettuare prenotazioni singole oppure collettive.
  + L’utente ha la possibilità di invitare altre persone se la sua prenotazione è collettiva.
  + Le strutture già registrate possono registrare nuovi campi.
  + Le strutture già registrate possono registrare nuovi spogliatoi.
  + Strutture non ancora registrate sulla piattaforma possono registrarsi.
* Could have
  + La piattaforma propone una serie di strutture disponibili e che rispettano i canoni indicati precedentemente dall’utente.
  + L’utente ha la possibilità di scegliere tra le strutture proposte in automatico dalla piattaforma.
  + L’utente ha la possibilità di disdire prenotazioni già effettuate entro 2 ore dall’evento.
  + La piattaforma distingue tra clienti standard e clienti convenzionati.
  + La piattaforma calcola il costo della prenotazione rispetto alla tipologia di cliente.
* Won’t have
  + La piattaforma informa l’utente se la struttura fornisce al cliente materiale tecnico oppure no.
  + Le strutture possono bandire utenti sgraditi.

Nella fase di specifica dei requisiti il team ha seguito la standard IEEE, dunque questi sono: