

**“ TOKYOGROUP - BARBER SHOP ”**

1. ***Introduzione***

Il nostro progetto di Ingegneria del software fa riferimento ad un gestionale che permette di gestire gli appuntamenti presso un negozio di parrucchieri.

È basato sulla creazione di due diverse tipologie di account che permettono di effettuare più azioni sul software, ovviamente entrambi dispongono di soluzioni per gestire l’account, come ad esempio l’eliminazione dell’account o il cambio password.

Il primo tipo di account è Gestore il quale è in grado di visualizzare gli appuntamenti a suo carico della settimana, creare e disdire un appuntamento.

Inoltre abbiamo pensato che ogni parrucchiere può offrire dei diversi servizi che una volta inseriti nel proprio account il cliente può usufruirne.

L’ultima funzionalità del gestore è che può leggere i messaggi inviati dai clienti.

L’altro account è Cliente che può prenotare un appuntamento selezionando oltre a giorno ed ora, il parrucchiere che più desidera e spuntare i vari servizi di cui ha bisogno.

Ha la possibilità di disdire l’appuntamento e di lasciare un messaggio ad un gestore.

1. ***Modello di processo***

Per la realizzazione del progetto il modello da noi utilizzato è quello Waterfall.

Questo approccio è basato su una sequenza lineare di fasi ben definite dove ogni fase deve essere completata prima di passare a quella successiva garantendo un avanzamento strutturato ed ordinato.

Le fasi da noi sviluppate sono: analisi dei requisiti, progettazione, implementazione, verifica e manutenzione.

Abbiamo definito i requisiti sulla base del modello MoSCoW:

| **MUST HAVE** | **SHOULD HAVE** | **COULD HAVE** | **WON’T** |
| --- | --- | --- | --- |
| Gestione appuntamenti | Cambio password | Recensioni su gestore | Notifiche |
| Invio messaggi | Eliminazione account |  | Interfaccia web |
| Login e registrazione | Modifica servizi offerti |  |  |
| Report settimanali |  |  |  |

1. ***Organizzazione del progetto***

Il team è composto da 4 persone:

* Mattia Curto
* Daniele Lazzari
* Andrea Roggeri
* Andrea Milani

Il lavoro è stato strutturato attraverso dei meeting in presenza nei quali ci siamo dedicati alla stesura dei requisiti e ad una parte della progettazione, dopo di che i meeting vengono effettuati online (usando Google Meet) per l’implementazione e testing del progetto.

1. ***Standard, linee guida, procedure***

Abbiamo seguito gli standard di Java in quanto il progetto è stato implementato su Eclipse.

Nel nostro progetto abbiamo utilizzato il database **H2** come soluzione di archiviazione dei dati, integrandolo tramite la dipendenza Maven specificata nel file pom.xml

La scelta del database H2 è stata fatta poiché l’abbiamo trovata particolarmente adatta per ambienti di sviluppo e test oltre che alla sua compatibilità con JDBC (Java Database Connectivity) che ci ha permesso di connettere velocemente il database.

1. ***Attività di gestione***

Le attività di gestione servono per gestire e controllare tutti i progressi durante lo sviluppo del progetto. Per quanto riguarda le attività di gestione ordinaria ogni giorno faremo una videochiamata in modo da allinearci sui vari progressi.

Ogni cambiamento da quanto deciso in origine verrà discusso facendo un meeting online dove ci saranno tutti i membri del team.

1. ***Rischi***

I rischi maggiori sono:

* Mancare la consegna finale del progetto causa impegni universitari e lavorativi di ciascun membro del team.
* Non riuscire a sviluppare un algoritmo abbastanza sofisticato che quindi non può essere utilizzato.

1. ***Personale***

* Curto Mattia (reviewer, tester, UML, documentazione)
* Lazzari Daniele (backend, frontend, UML, documentazione)
* Andrea Roggeri (backend, progettista database, UML, documentazione)
* Andrea Milani (reviewer, tester, UML, documentazione)

1. ***Metodi e tecniche***

Utilizziamo UML e i diversi tipi di diagrammi per la modellazione del programma.

* USE CASE DIAGRAM: per i casi d’uso e gli attori
* CLASS DIAGRAM: per attributi e metodi delle classi
* SEQUENCE DIAGRAM: per una modellazione sequenziale e un’ordine di esecuzione
* STATE MACHINE: per i diversi stati di un oggetto all'interno di un sistema e le transizioni che avvengono tra questi stati in risposta a eventi o condizioni.
* COMUNICATION DIAGRAM: illustra le interazioni tra gli oggetti o componenti di un sistema, concentrandosi sui messaggi scambiati tra di essi per raggiungere un obiettivo specifico.
* ACTIVITY DIAGRAM: per descrivere il flusso di lavoro o il processo, evidenziando l'ordine delle attività e le decisioni prese lungo il percorso.
* PACKAGE DIAGRAM: per mostrare classi, interfacce o componenti sono raggruppati in diversi package.

1. ***Garanzia di qualità***

Il programma punta a garantire una qualità di livello dello standard ISO 9001.

Inoltre, per garantire la sicurezza degli utenti, sono state adottate librerie moderne, ampiamente utilizzate e regolarmente aggiornate, riducendo così il rischio di vulnerabilità.

Cercheremo di garantire che il prodotto sia utilizzabile, corretto ed affidabile.

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

* Database embedded H2
* Backend sviluppato in Eclipse con JAVA
* Frontend sviluppato in Eclipse con SWING

Ogni membro del team partecipa attivamente allo sviluppo dei vari componenti.

1. ***Risorse***

Per lo sviluppo del progetto ci saranno:

* Team: formato da Mattia Curto, Daniele Lazzari, Andrea Roggeri e Andrea Milani
* GitHub: verrà svolta la condivisione del lavoro
* StarUML: usato per realizzare i diversi diagrammi UML
* Eclipse: verrà usato come ambiente di sviluppo
* Visual Studio Code: usato per modifiche i linea
* SQLite: usato per la gestione del database
* Google Meet: usato per le riunioni del Team
* Log4J: libreria di Java che ci permette di tenere traccia di tutti i LOG
* JavaDOC: tool di JAVA per la documentazione

1. ***Budget***

Il nostro budget non è misurato in termini economici ma è relativo alla quantità di tempo che dedicheremo al progetto.

Abbiamo concordato che ogni membro impegnerà un minimo di 40 ore con la possibilità di superare questa soglia.

1. ***Cambiamenti***

La manutenzione pianificata per il futuro comprenderà tutte le quattro principali tipologie previste dalla teoria dei sistemi software:

- Correzione di errori: finalizzata alla risoluzione di bug o malfunzionamenti individuati nel software.

- Adattamento ai cambiamenti dell’ambiente operativo, al fine di garantire la compatibilità con nuove piattaforme tecnologiche, normative o requisiti specifici del contesto.

- Perfezionamento del sistema: basato sia sulle richieste degli utenti finali sia su iniziative del team volte a migliorare le prestazioni, l’efficienza o l’esperienza d’uso.

- Prevenzione: orientata a migliorare la manutenibilità del software, ad esempio mediante il costante aggiornamento della documentazione tecnica e funzionale.

Le modifiche proposte saranno preventivamente analizzate e discusse dal team di progetto, garantendo un processo decisionale condiviso e consapevole. Inoltre, per ogni nuova versione del sistema, verrà rilasciata la documentazione aggiornata, in conformità con le buone pratiche di gestione del ciclo di vita del software.

1. ***Consegna***

Per la consegna il progetto verrà condiviso in un repository GitHub, dove caricheremo la documentazione e il codice relativo all’applicazione sviluppata.

Indicativamente la consegna è fissata per il 04/01/2025 con possibilità di manutenzione fino

alla data dell’orale da concordare con il professore.