MatchPoint

Project Plan V. 1.0

Sommario

[1. Introduzione 2](#_Toc181629583)

[2. Modello di processo 2](#_Toc181629584)

[3. Organizzazione del progetto 2](#_Toc181629585)

[4. Norme, linee guida e procedure 2](#_Toc181629586)

[5. Attività di gestione 2](#_Toc181629587)

[6. Rischi 2](#_Toc181629588)

[7. Personale 2](#_Toc181629589)

[8. Metodi e tecniche 2](#_Toc181629590)

[9. Garanzie di qualità 2](#_Toc181629591)

[10. Pacchetti di lavoro 2](#_Toc181629592)

[11. Risorse 2](#_Toc181629593)

[12. Budget e programma 2](#_Toc181629594)

[13. Cambiamenti 2](#_Toc181629595)

[14. Consegna 2](#_Toc181629596)

## Introduzione

L’obiettivo del progetto “MatchPoint” è sviluppare un’applicazione web per la prenotazione e la gestione dei campi da gioco nei centri sportivi in Italia. Ciò nasce dall’esigenza di gruppi di amici e appassionati di sport che sono in difficoltà a trovare campi e terreni di gioco vicini e a prezzi accessibili.

I gestori dei centri sportivi possono registrarsi ed inserire i campi che hanno a disposizione (sport, terreno, dimensioni, illuminazione, disponibilità oraria, costi e servizi), mentre gli utenti (giocatori) possono prenotare i campi a fasce orarie e gli eventuali servizi abbinati.

Il progetto sarà sviluppato da Filippo Bonfanti, Matteo Colombi e Luca Rossi nell’ambito del corso di Ingegneria del software (A.A. 2024/2025 Proff. Gargantini e Bonfanti).

## Modello di processo

Per questo progetto si è deciso di utilizzare un processo di tipo SCRUM; poiché non è possibile individuare un product owner ed uno SCRUM master, questi due ruoli sono visti come “ipotetici” e le loro esigenze verranno immaginate dal team di sviluppo.

Scrum è un metodo Agile che gestisce progetti in contesti difficili da pianificare in anticipo. Adotta un approccio iterativo e incrementale, focalizzandosi su trasparenza, ispezione e adattamento per ottimizzare prevedibilità e gestione del rischio.   
Il progetto viene suddiviso in brevi cicli di lavoro chiamati Sprint (durata 1-4 settimane), ciascuno dei quali produce un incremento del software. Il team Scrum include il Product Owner (rappresentante degli stakeholder), il Development Team (autonomo) e lo Scrum Master (facilita il team e rimuove ostacoli).  
Il Product Owner crea e gestisce il product backlog, una lista di priorità. Gli Sprint sono pianificati all'inizio, fissando obiettivi e tempi. Durante uno Sprint, gli obiettivi non cambiano, e ogni giorno il team si riunisce nel Daily Scrum. Alla fine di ogni Sprint si svolgono la Sprint Review e la Sprint Retrospective. Il ciclo si ripete fino a completare il backlog, esaurire il budget o completare il prodotto.

Le milestones di questo progetto sono le seguenti:

* **Analisi dei requisiti**
  + Definizione dettagliata delle funzionalità, con focus su esigenze degli utenti e requisiti dei gestori dei centri sportivi.
* **Design dell'architettura** 
  + Progettazione dell'architettura software, compresa la definizione del database e delle interfacce principali.
* **Sviluppo prototipo funzionale** 
  + Implementazione di un prototipo minimo per il testing delle funzionalità chiave come la registrazione degli utenti, la prenotazione dei campi e la gestione dei profili.
* **Testing e validazione del prototipo**
  + Esecuzione di test di usabilità, prestazioni e funzionalità sul prototipo per identificare eventuali problemi e miglioramenti.
* **Sviluppo delle funzionalità complete** 
  + Implementazione delle funzionalità aggiuntive: gestione degli orari, personalizzazione dei campi, pagamento e notifiche.
* **Testing finale e miglioramento**
  + Test completi per garantire che l'app soddisfi i requisiti di qualità e sia pronta per la distribuzione.
* **Rilascio della versione 1.0** 
  + Lancio ufficiale dell'applicazione.

I percorsi critici sono la gestione della base di dati e del login oltre che la parte grafica del progetto.

## Organizzazione del progetto

Il team di sviluppo è composto da Filippo Bonfanti, Matteo Colombi e Luca Rossi, i membri collaboreranno attivamente a tutte le parti del progetto, in particolare ogni membro sarà responsabile di alcune parti di esso:

* **Filippo Bonfanti:**
* **Matteo Colombi:**
* **Luca Rossi:**

## Norme, linee guida e procedure

La scrittura della documentazione è un’attività che si svolge dividendola in parti uguali e assegnandole a ciascun sviluppatore del team; una volta conclusa la fase di scrittura viene fatta una revisione completa da parte di tutti i membri in modo da segnalare eventuali errori o imprecisioni.  
Inoltre la documentazione viene aggiornata ogni qual volta vi sia un cambiamento, segnando anche la creazione di un’eventuale nuova versione, in modo tale che tutto rimanga tracciato e in caso di errori sarà sempre possibile tornare a versioni precedenti.

In particolare, il rapporto tra la fase del modello e del design sarà governato dal **Model Driven Architecture (MDA)**: di solito negli approcci tradizionali in cui si fa un modello durante la fase di requisiti e di design e questa si traduce in codice, qualsiasi evoluzione viene fatta cambiando il codice, non il modello, ma così facendo il modello diventa presto obsoleto. MDA invece pensa ad aggiornare di volta in volta il modello e da questo si arriva al codice. Di conseguenza la manutenzione va fatta prima sul modello.

Il progetto seguirà le convenzioni per la programmazione Java come definite da Oracle.

## Attività di gestione

Le attività di gestione del progetto sono guidate da obiettivi specifici e priorità stabilite per garantire il rispetto dei requisiti, dei tempi e dei costi pianificati.

In primis bisogna gestire il **monitoraggio e reporting**, il team impegna a presentare relazioni periodiche sullo stato di avanzamento del progetto. Questi rapporti, redatti tendenzialmente alla fine di ogni Sprint, forniranno un quadro complessivo dei progressi, delle sfide incontrate e dei risultati raggiunti rispetto agli obiettivi prefissati. La presenza regolare di questi report permetterà al team di tenere sotto controllo eventuali ritardi o scostamenti rispetto al piano originale, facilitando interventi correttivi e tempestivi.

Un’altra attività molto importante è il **bilanciamento dei requisiti**: durante lo sviluppo, sarà necessario bilanciare i requisiti funzionali e non funzionali del sistema. Ciò comporta la gestione delle priorità assegnate a ciascun requisito, in base al modello MoSCoW, e la verifica continua che le funzionalità chiave siano sviluppate e testate secondo le aspettative degli utenti. Eventuali modifiche ai requisiti saranno valutate e inserite nei successivi Sprint solo se coerenti con gli obiettivi e le risorse disponibili. Di seguito viene approfondito il modello **MoSCoW**:

* **Must haves**: requisiti prioritari, si devono realizzare per rendere il sistema accettabile.
* **Should haves**: requisiti non obbligatori, ma altamente desiderabili.
* **Could haves**: requisiti che, se il tempo lo consente, verranno realizzati.
* **Won't haves**: non saranno realizzati, ma sono registrati. Essi potranno essere presi in considerazione in futuro.

La **gestione dei tempi** è sicuramente un aspetto fondamentale per la buona riuscita del progetto. Il rispetto delle scadenze è fondamentale per mantenere il progetto nei limiti stabiliti. Durante le riunioni di Sprint Planning, il team definirà gli obiettivi da raggiungere entro il termine di ogni Sprint, pianificando dettagliatamente le attività necessarie e stimando accuratamente i tempi. In caso di ritardi imprevisti, i membri rivedranno il piano per riadattare le priorità e garantire che le funzionalità essenziali siano completate entro le scadenze previste.

Un altro aspetto su cui il team si focalizzerà molto è sulla **gestione della qualità**, che verrà meglio approfondita nel paragrafo 9.

Infine è fondamentale mantenere un’ottima **comunicazione interna**. Una comunicazione efficace tra i membri del team è essenziale per una gestione efficiente del progetto. Saranno organizzate riunioni regolari, come il Daily Scrum, per aggiornamenti giornalieri, e revisioni più estese per fare il punto della situazione al termine di ogni Sprint. In caso di problemi significativi o modifiche ai requisiti, si potrà aprire un issue su Git, in modo tale da informare tutti gli altri membri del team, assicurando che ogni decisione sia condivisa e compresa.

## Rischi

Durante il progetto “MatchPoint” possono emergere diversi rischi che dovranno essere identificati il prima possibile per minimizzare gli impatti negativi che potrebbero portare. I principali rischi includono:

* **Mancanza di informazioni critiche**: potrebbero verificarsi situazioni in cui informazioni essenziali per l’analisi dei requisiti o per il testing risultano incomplete o mancanti, aumentando il rischio di non soddisfare le esigenze degli utenti.
* **Problemi tecnici imprevisti**: bug critici, fallimenti di integrazione o incompatibilità con sistemi esterni possono rallentare il processo di sviluppo.

Per affrontare questi rischi, si pianificano azioni di mitigazione, come la creazione di un calendario con scadenze anticipate per le consegne, il monitoraggio continuo delle risorse disponibili e l’aggiornamento costante della documentazione. Inoltre, si terranno sessioni di revisione periodica per identificare tempestivamente eventuali problematiche.

## Personale

Il progetto richiederà diverse quantità di personale con competenze differenti in base alla fase in corso. Le risorse sono distribuite come segue:

* **Analisi dei requisiti e design**: questa fase richiede competenze in ingegneria dei requisiti, analisi di sistema e progettazione software. Saranno impiegati membri del team con esperienza nell'analisi e design.
* **Sviluppo del prototipo e funzionalità complete**: necessità di sviluppatori con competenze avanzate in linguaggi di programmazione web (come JavaScript, HTML, CSS) e backend (Java) e di gestione di databases (MongoDB). Durante questa fase, tutti i membri saranno attivi, ciascuno con responsabilità specifiche su moduli e funzionalità.
* **Testing e qualità**: richiede competenze in testing manuale e automatizzato, e tecniche di quality assurance. Un membro del team sarà assegnato alla supervisione della qualità del software, mentre gli altri supporteranno i test periodici e le revisioni del codice.
* **Rilascio e manutenzione**: include competenze di deployment e gestione di sistema per il rilascio dell'applicazione. Saranno necessari membri del team per configurare e monitorare il lancio del software.

Ogni fase sarà eseguita garantendo una collaborazione fluida tra i membri del team e sfruttando competenze complementari per ottimizzare i tempi e la qualità complessiva del progetto.

## Metodi e tecniche

L’obiettivo è garantire che ogni fase del progetto sia supportata da processi strutturati e strumenti adeguati alla produzione e la gestione efficace del software.

Durante **l’ingegneria dei requisiti**, verranno utilizzati metodi di raccolta e analisi dei requisiti come interviste, questionari e brainstorming. I diagrammi UML (Unified Modeling Language) saranno impiegati per descrivere funzionalità e flussi di lavoro del sistema.  
La documentazione prodotta in questa fase verrà utilizzata come base per le successive attività di progettazione.

Per la **progettazione**, verranno utilizzate tecniche di modellazione UML, inclusi diagrammi di classe, sequenza e attività per definire le interazioni tra i componenti. Verranno inoltre applicate tecniche di progettazione modulare per facilitare l'integrazione e la manutenzione del sistema. Ogni componente sarà documentato per dettagliare le interfacce, le dipendenze e i requisiti specifici, utilizzando strumenti come Visio o Lucidchart.

Durante l’**implementazione**, saranno utilizzati metodi di sviluppo incrementale per consentire il rilascio di versioni intermedie, fornendo ai team di test componenti già parzialmente funzionanti. Ogni rilascio sarà accompagnato da documentazione di codice, incluse annotazioni dettagliate, diagrammi di flusso, e informazioni sull’architettura adottata. Per la gestione della versione, verrà utilizzato un sistema di controllo della versione distribuito (es. Git), che consentirà di tracciare le modifiche al codice, gestire branche e versioni, e facilitare il lavoro collaborativo.

La fase di **test** seguirà una metodologia sia di testing unitario che di integrazione, usando strumenti automatizzati ove possibile (es. JUnit per Java o PyTest per Python) per garantire una verifica rapida e affidabile del codice. Verranno effettuati anche test di regressione a ogni aggiornamento per assicurarsi che le nuove funzionalità non compromettano quelle esistenti. Tutti i casi di test e i risultati saranno documentati per tenere traccia delle funzionalità testate e dei bug riscontrati.

Per garantire **coerenza** e **tracciabilità** durante lo sviluppo, verrà implementato un rigoroso sistema di controllo di versione tramite Git, con pratiche di commit regolari e messaggi descrittivi. Le configurazioni software saranno gestite in repository separati per gli ambienti di sviluppo, test e produzione, utilizzando strumenti come Docker per facilitare la coerenza tra le installazioni.

La **documentazione** tecnica sarà prodotta e mantenuta durante tutto il ciclo di vita del progetto, in modo che rimanga aggiornata e rifletta le modifiche apportate in ogni fase. Verrà adottato uno standard di documentazione coerente (come l’uso di Markdown o LaTeX), e il processo di revisione sarà pianificato per garantire che le specifiche tecniche siano accuratamente descritte e facilmente consultabili.

## Garanzie di qualità

## Per assicurare la qualità del software sviluppato in MatchPoint, verranno implementati controlli sistematici e test periodici, con l’obiettivo di mantenere il processo di sviluppo in linea con gli standard di qualità stabiliti.

Un membro del team si occuperà del **Quality Assurance (QA)** e sarà incaricato di definire e coordinare i test e le verifiche periodiche durante lo sviluppo. L’intero team sarà comunque responsabile di scrivere codice conforme alle linee guida e convenzioni stabilite, segnalando prontamente eventuali anomalie o incongruenze. Ogni membro del team contribuirà alle attività di **testing manuale e automatizzato** per verificare che il software rispetti i requisiti di qualità.

Inoltre, verranno rispettati gli standard di **usabilità** e **accessibilità** per garantire che l'interfaccia sia semplice e intuitiva per gli utenti finali (giocatori e gestori) e che i tempi di risposta dell'applicazione siano adeguati. Saranno definiti obiettivi specifici come i tassi di difetti accettabili e l’uso del software su diversi browser e dispositivi.

**Code review** periodiche verranno eseguite per individuare e correggere errori di sintassi e altri bug, mentre **test unitari** e **test di integrazione** saranno eseguiti per ogni modulo sviluppato, in modo da verificare l’interazione corretta tra i vari componenti, sia individualmente che integrati nel sistema complessivo.

Alla conclusione dello sviluppo, verrà eseguita una fase finale di **testing** per assicurare il corretto funzionamento dell’applicazione. Infine, la **documentazione** sarà completata e consolidata, incorporando tutte le informazioni raccolte e aggiornate durante il processo di sviluppo.

## Pacchetti di lavoro

Il progetto è stato suddiviso in alcuni **pacchetti** di lavoro come segue:

1. Stesura del project plan e gestione del progetto
2. Analisi dei requisiti
3. Design del software
4. Implementazione
   1. Sviluppo classi
   2. Sviluppo DB
   3. Sviluppo backend
   4. Sviluppo GUI
5. Testing
6. Manutenzione

Per ogni pacchetto di lavoro saranno previsti uno o più **sprint** della durata di 1 o 2 settimane; prima di ogni sprint, la squadra decide lo sprint backlog (obiettivi da raggiungere al termine dello sprint) e al termine dello sprint effettua un’analisi dei risultati raggiunti (sprint retrospective). Al termine di ogni sprint i risultati vengono aggiunti al product backlog.

## Risorse

Per quanto riguarda l’aspetto delle **risorse umane** al progetto lavorano i tre studenti precedentemente citati, che si sono distribuiti in maniera equa il lavoro da svolgere, in particolare:

* **Filippo Bonfanti**
* **Matteo Colombi**
* **Luca Rossi**

Le **risorse tecnologiche** utilizzate sono le seguenti:

* Eclipse per la parte di sviluppo del codice
* MongoDB / … per il DataBase
* …. Per la parte di GUI
* Papyrus per la modellazione ed i grafici UML
* GitHub per il versioning e la gestione collaborativa del repository
* … per il testing

Vengono inoltre impiegate delle **risorse materiali**, ovvero i portatili dei tre studenti ed i loro tablet, oltre che i testi del corso di ingegneria del software (software engineering e UML@Classroom)

Le **risorse economiche** non rientrano in questo project plan in quanto vengono utilizzate piattaforme e strumenti open source o comunque gratuite.

## Budget e programma

## Cambiamenti

## Consegna