Università degli Studi di Bergamo
Dipartimento di Ingegneria
Corso di Laurea in Ingegneria Informatica

Anno Accademico 2025/2026

PROJECT PLAN

Ingegneria del Software

The Scouting Suite

Brivio Andrea	Matricola: 1089359
Fischetti Alessandro	Matricola: 1080491
Pesenti Lorenzo	Matricola: 1072635

Versione 2.0
9 Dicembre 2025

Cronologia Versioni

Ver.	Data	Descrizione
1.0	28/10/2025	Prima stesura del Project Plan
1.1	30/10/2025	Definizione obiettivi e sezione Introduction
1.2	02/11/2025	Definizione Process Model e metodologia SCRUM
1.3	05/11/2025	Definizione ruoli del team e struttura organizzativa
1.4	08/11/2025	Aggiunta Standards, Guidelines e Management Activities
1.5	11/11/2025	Risk Management e Staffing
1.6	14/11/2025	Quality Assurance, Methods and Techniques
1.7	17/11/2025	Definizione Workpackages, Resources e Schedule
1.8	19/11/2025	Aggiornamento documentazione
1.9	20/11/2025	Revisione formale
2.0	09/12/2025	Revisione finale dei contenuti pre-consegna

Tabella 1: Cronologia delle versioni del Project Plan

1 Introduction

Il progetto *The Scouting Suite* ha l'obiettivo di realizzare un sistema software avanzato per la gestione strutturata delle informazioni relative allo scouting calcistico. L'applicativo è concepito per consentire la raccolta, l'organizzazione e l'analisi dei dati relativi ai giocatori, offrendo agli operatori del settore uno strumento di supporto decisionale efficace.

Il contesto operativo è quello dell'analisi sportiva (Sports Analytics), un settore in cui la gestione del dato è divenuta critica per le decisioni strategiche. Il perimetro del progetto comprende la progettazione di un database relazionale ottimizzato, un'architettura software modulare e l'adozione di metodologie di sviluppo che assicurino manutenibilità ed estensibilità.

Gli obiettivi primari sono:

- **Data Management:** Progettazione di un database robusto per le attività di scouting.
- **Sviluppo Agile:** Adozione del framework SCRUM per uno sviluppo iterativo.
- **Documentazione:** Produzione di documentazione tecnica formale e coerente.
- **Prototipazione:** Realizzazione di un prototipo funzionante a dimostrazione dell'architettura.

Il progetto è sviluppato nell'ambito del corso di *Ingegneria del Software* presso l'Università degli Studi di Bergamo.

2 Process Model

Per lo sviluppo di *The Scouting Suite* è stato adottato il framework **SCRUM**, un modello Agile che favorisce lo sviluppo iterativo e incrementale. Questa scelta permette di gestire efficacemente i cambiamenti nei requisiti e di fornire valore tangibile in tempi ridotti.

A differenza del modello a cascata, il focus è sulla consegna frequente di incrementi software funzionanti. Questo approccio riduce il tempo dedicato alla pianificazione predittiva a lungo termine in favore di una pianificazione adattiva.

Il ciclo di vita è scandito da *Sprint*, iterazioni di durata prefissata (time-boxed) pari a una settimana. Ogni Sprint include:

- **Sprint Planning:** Definizione degli obiettivi dell'iterazione.
- **Development Testing:** Scrittura del codice e test di unità (JUnit) continui.
- **Sprint Review:** Verifica del lavoro svolto con il Product Owner.
- **Sprint Retrospective:** Analisi del processo per il miglioramento continuo.

Il processo è supportato da artefatti e strumenti quali il Product Backlog, la Kanban Board per la visualizzazione del flusso e i Weekly Scrum per il coordinamento. La documentazione, redatta in LaTeX, viene aggiornata in parallelo allo sviluppo, coprendo le Milestones di Gestione, Requisiti, Design, Testing e Manutenzione.

3 Organization of the Project

Il team di sviluppo opera secondo i ruoli definiti da SCRUM ed è composto da:

- **Brivio Andrea**
- **Fischetti Alessandro**
- **Pesenti Lorenzo**

La collaborazione avviene prevalentemente in presenza per massimizzare la velocità di comunicazione, supportata dall'uso di GitHub per il controllo di versione distribuito. Il progetto è supervisionato dal Prof. Angelo Gargantini e dalla ricercatrice Silvia Bonfanti.

4 Standards, Guidelines, Procedures

Per garantire la qualità e la leggibilità del codice, il team adotta Standards rigorosi:

- **Coding Standard:** Adozione delle *Oracle Java Code Conventions* (es. PascalCase per le classi, camelCase per metodi e variabili).
- **Build Automation:** Utilizzo di **Maven** per la gestione delle dipendenze e l'automazione del ciclo di vita della build.
- **Documentazione del Codice:** Utilizzo sistematico di **Javadoc**. La documentazione è scritta contestualmente al codice e revisionata dagli altri membri del team (Peer Review).

Procedure operative:

- **Version Control:** Strategia di branching su GitHub.
- **Code Review:** Ogni Pull Request deve essere approvata da almeno un altro membro del team prima del merge sul branch principale.

5 Management Activities

Monitoraggio del progresso

Il monitoraggio avviene tramite eventi SCRUM adattati alle esigenze accademiche. Al posto del Daily Scrum, il team effettua *Weekly Scrums* approfonditi per pianificare la settimana, integrati da brevi check giornalieri asincroni. La *Sprint Review* al termine di ogni iterazione permette di validare le funzionalità completate rispetto ai criteri di accettazione.

Prioritizzazione

Il Product Backlog è ordinato secondo la tecnica **MoSCoW** (Must have, Should have, Could have, Won't have), garantendo che le funzionalità critiche siano sviluppate per prime.

Gestione delle problematiche

L'approccio alla risoluzione dei problemi è collaborativo. Gli impedimenti tecnici vengono tracciati come *Issues* su GitHub. Qualora un blocco non sia risolvibile internamente, viene scalato ai docenti di riferimento.

6 Risks Potential

La gestione dei rischi è proattiva. Di seguito i principali rischi identificati e le relative strategie di mitigazione.

Rischio	Mitigazione
Indisponibilità membri	Pianificazione con margini (slack), condivisione della conoscenza (pair programming).
Requisiti ambigui	Raffinamento continuo del Backlog con il Product Owner.
Difficoltà tecniche	Adozione di spike (ricerca) preliminari, suddivisione in task granulari.
Problemi di comunicazione	Canali diretti, uso rigoroso della Kanban board.

Tabella 2: Analisi dei rischi e mitigazione

In caso di ritardi significativi, il team applicherà la riallocazione delle risorse o la riduzione dello scope (funzionalità "Could have") per rispettare la data di consegna.

7 Staffing

I ruoli sono assegnati per valorizzare le competenze individuali pur mantenendo la flessibilità (cross-functionality) richiesta da SCRUM.

- **Product Owner (Brivio Andrea):** Responsabile della visione del prodotto, gestione del Backlog e definizione delle priorità.
- **Scrum Master (Fischetti Alessandro):** Responsabile del processo, rimozione degli ostacoli e facilitazione dei meeting.
- **Development Team (Tutti):**
  - *Brivio Andrea*: Focus su Logica di Business e Database.
  - *Fischetti Alessandro*: Focus su Architettura e UI.
  - *Pesenti Lorenzo*: Focus su Testing (QA) e Integrazione.

Il carico di lavoro è distribuito uniformemente da Ottobre a Gennaio.

8 Methods and Techniques

Il progetto utilizza metodi standardizzati per ogni fase del ciclo di vita:

Attività	**Tecnica / Metodo**
Requisiti	Casi d'uso UML, User Stories, Interviste.
Prioritizzazione	Metodo MoSCoW.
Design	Modellazione UML (Class, Sequence, State diagrams) con Papyrus.
Implementazione	Programmazione a Oggetti (Java), Sviluppo incrementale.
Testing	JUnit per unit test, Integrazione continua.

Tabella 3: Metodologie adottate

9 Quality Assurance

La garanzia della qualità è integrata nel processo (Built-in Quality). Le attività principali includono:

- **Static Analysis:** Revisione continua del codice e dei diagrammi UML per verificare l'aderenza agli standard.
- **Testing Automatizzato:** Esecuzione di test di unità JUnit prima di ogni commit significativo.
- **Gestione Difetti:** Tracciamento e risoluzione sistematica dei bug tramite GitHub Issues.
- **Metriche:** Monitoraggio della copertura dei test e della complessità del codice.

L'obiettivo è rispettare i criteri di qualità ISO/IEC 9126, con particolare attenzione all'affidabilità e alla manutenibilità.

10 Workpackages

Il lavoro è strutturato in pacchetti logici (Work Packages) assegnati per competenza ma svolti collaborativamente:

- **WP1 - Project Management:** Gestione Backlog, coordinamento Sprint (Resp: Brivio, Fischetti).
- **WP2 - Analysis & Design:** Modellazione UML e specifica requisiti (Team).
- **WP3 - Development:** Implementazione Core, DB e UI (Team).
- **WP4 - Testing:** Unit testing, System testing (Resp: Pesenti).
- **WP5 - Documentation:** Stesura documenti di progetto e Javadoc (Team).

11 Resources

Le risorse necessarie per lo sviluppo sono principalmente software e strumenti di collaborazione.

Ambito	Risorse
Gestione	GitHub (Project Board, Issues), Overleaf (LaTeX).
Design	Eclipse Papyrus (UML).
Sviluppo	JDK 17+, Eclipse IDE/IntelliJ, Maven, Python (Data scraping).
Dati	Dataset proprietario estratto da Fbref (stagione 2024/2025).
Testing	JUnit 5.

Tabella 4: Risorse Hardware e Software

Gestione Dati

Per popolare il database a costo zero, è stato sviluppato uno script Python per lo scraping etico dei dati da *Fbref*. I dati sono stati normalizzati per gestire incongruenze (es. duplicati per trasferimenti invernali) e raffinati per l'importazione nel sistema Java.

12 Budget and Schedule

Dato l'approccio Agile/SCRUM adottato, la pianificazione non è rigida ma adattiva. Tuttavia, per garantire il rispetto della scadenza accademica, è stata definita una roadmap che funge da guida per gli Sprint.

Pianificazione delle Attività

La pianificazione avviene su due livelli:

- **Roadmap di Progetto:** Definisce le milestone principali (Analisi, Architettura, MVP, Release Finale).
- **Sprint Planning:** All'inizio di ogni settimana, il team seleziona le User Stories con priorità maggiore dal Backlog da implementare nello Sprint corrente.

Strumenti di Programmazione e PERT

Per la definizione delle dipendenze critiche iniziali (soprattutto nella fase di architettura), viene utilizzato un diagramma **PERT** (Program Evaluation and Review Technique) illustrato in Figura 1. Questo strumento permette di visualizzare il percorso critico delle attività propedeutiche che deve precedere lo sviluppo del layer di persistenza. Per la gestione operativa quotidiana, il team utilizza una **Kanban Board** su GitHub Projects.

Diagramma PERT delle Dipendenze

Il seguente grafico PERT illustra le dipendenze logiche tra le attività. Il percorso evidenziato in rosso rappresenta il **Critical Path**: un ritardo in queste attività comporterebbe uno slittamento della consegna finale.

Legenda delle dipendenze principali:

- L'attività **E (Data Import)** non può iniziare finché non sono completi lo **Scraping (B)** e la struttura del **Database (C)**.
- L'implementazione dell'interfaccia **(G)** richiede che la **Logica (F)** sia funzionante e che il **Design (D)** sia definito.

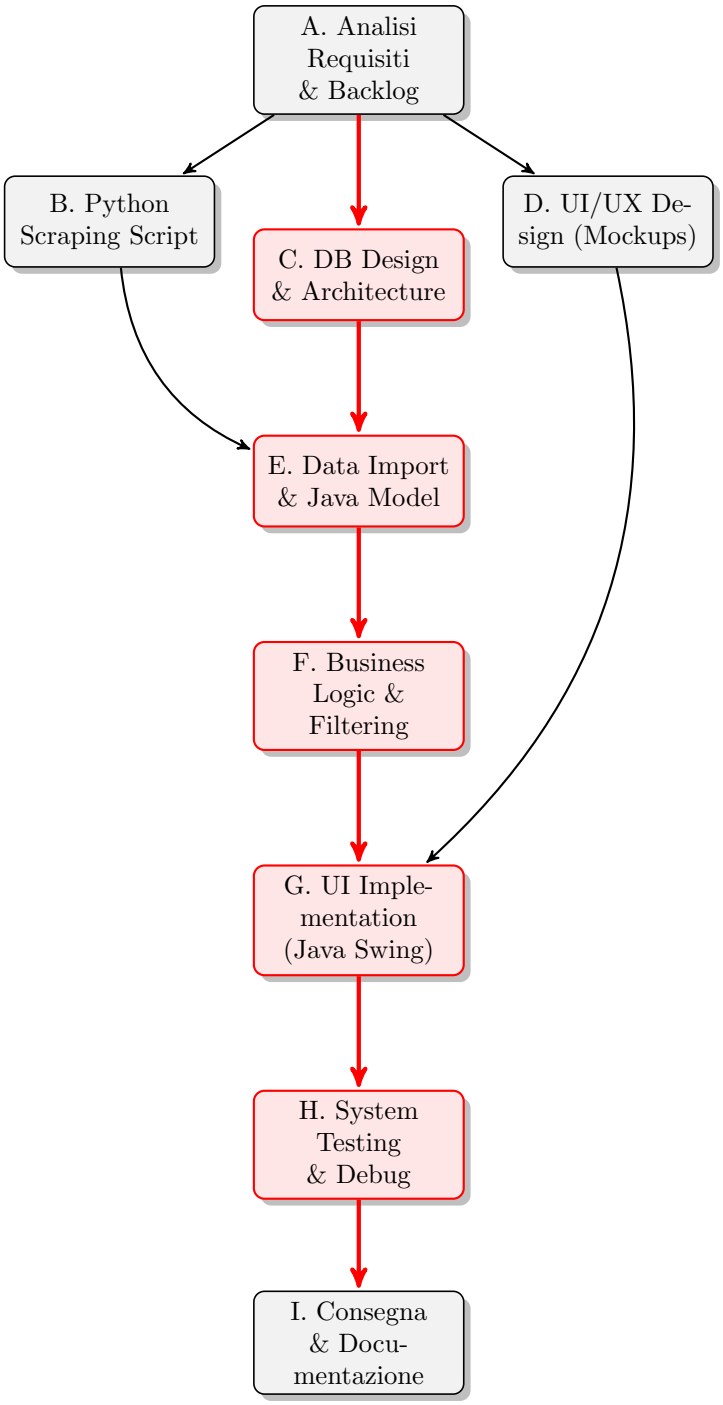

Figura 1: Diagramma PERT del progetto Scouting Suite. In rosso il Percorso Critico.

Tracciamento delle Spese e Risorse

Il progetto non presenta costi monetari diretti (budget finanziario nullo). La risorsa principale è il **tempo** (effort). Il tracciamento delle spese avviene monitorando le **ore/uomo** dedicate a ciascun Work Package. Ogni membro del team registra l'impegno profuso, permettendo di calcolare il "costo" effettivo dello sviluppo in termini di risorse umane.

Visibilità e Misure di Progresso

Per garantire trasparenza e misurabilità (Visibility of Progress):

- **Burndown Chart:** Utilizzato per visualizzare il lavoro rimanente rispetto al tempo a disposizione nello Sprint.
- **Velocity:** Misura della quantità di lavoro (Story Points) che il team riesce a completare in uno Sprint, utilizzata per stimare le iterazioni future.
- **Sprint Review:** Momento formale di ispezione dell'incremento prodotto.

Schedule (Roadmap degli Sprint)

A differenza del modello a cascata, le fasi di Design, Implementazione e Test si sovrappongono all'interno degli Sprint. Di seguito la pianificazione temporale effettiva:

Periodo	Fase / Sprint	Obiettivi e Attività Principali
Dic (W2)	**Sprint 0** (Inception)	Analisi del dominio, Scraping dati, Setup ambiente, Definizione Product Backlog iniziale.
Dic (W3-W4)	**Sprint 1** (Core)	Architettura UML, Design del Database, Implementazione layer Model e Persistence.
Gen (W1)	**Sprint 2** (Features)	Logica di Business (Scouting Algorithm), Implementazione Controller, Prime interfacce utente.
Gen (W2)	**Sprint 3** (Refine)	Completamento UI, Integrazione funzionalità avanzate, Refactoring del codice.
Gen (W3)	**Sprint 4** (Finalize)	Testing di sistema, Correzione Bug, Finalizzazione documentazione, Preparazione presentazione.

Tabella 5: Schedule del progetto organizzata per Sprint

13 Changes

La gestione delle modifiche ai requisiti o al codice è automatizzata tramite **GitHub**. Ogni modifica sostanziale segue un flusso strutturato:

1. Apertura di una *Issue* per descrivere la richiesta di modifica o il bug.
2. Creazione di un *Branch* dedicato.
3. Implementazione e Commit.
4. *Pull Request* e revisione da parte del team.
5. Merge sul ramo principale (Master/Main).

Questo garantisce tracciabilità totale di ogni cambiamento avvenuto durante il ciclo di vita del software.

14 Delivery

Il rilascio finale (Delivery) comprende:

- **Documentazione:** File PDF completo generato da LaTeX, contenente tutte le sezioni di Project Management, Requirements, Design, Testing e Maintenance.
- **Codice Sorgente:** Repository GitHub accessibile, pulito e commentato (Javadoc).
- **Eseguibile:** File `.jar` o istruzioni per la build del progetto funzionante.
- **Dati:** Script di popolamento del database o dump SQL per ricreare l'ambiente di test.

La consegna avverrà in due fasi: 1. Consegna della documentazione completa (Deadline: metà Dicembre). 2. Consegna del codice e demo del prototipo funzionante (Deadline: Gennaio, pre-appello).