Progetto di Ingegneria del Software 2025/26

Università Ca' Foscari Venezia

Piano di progetto

EcoGroup

21/10/2025

Document Informations

NomeProgetto	Acronimo	
Deliverable	Piano di progetto	
Data di Consegna	21 / 10 / 2025	
Team Leader	Enrico Sforza	902600@stud.unive.it
Team members	Enrico Rampazzo 901141@stud.unive.it Alberto Ferragosti 895936@stud.unive.it	

Document History

Version	Issue Date	Stage	Changes	Contributors
1.0	19 / 10 / 2025	Draft	Completati parzialmente i punti 1, 2, 3, 4	902600
1.1	20 / 10 / 2025	Draft	Completati i primi 4 punti	901141
2.0	21 / 10 / 2025	Final	Completata la sezione 5	895936
3.0	25/01/2026	Final	Revisione pre consegna	895936

Indice

1. Introduzione

1.1. Overview

EcoApp si propone come uno strumento di **educazione e cambiamento comportamentale**, concepito per sensibilizzare e formare le nuove generazioni su pratiche quotidiane sostenibili attraverso un approccio ludico e motivazionale (gamification).

Gli **obiettivi principali** sono:

1. **Educare alla sostenibilità ambientale**
   Promuovere la comprensione e l'adozione di comportamenti sostenibili nelle aree chiave dell'impatto ambientale personale: rifiuti, mobilità, energia, alimentazione e consumo consapevole.
2. **Trasformare la consapevolezza in azione concreta**
   Favorire il passaggio dalla semplice conoscenza teorica all'adozione di abitudini quotidiane sostenibili, misurabili e replicabili nel tempo.
3. **Rendere tangibile l'impatto personale**
   Offrire agli utenti una rappresentazione chiara, visiva e quantitativa del proprio contributo alla riduzione dell'impronta ambientale (es. kg di CO_2 evitata), stimolando senso di responsabilità e motivazione.
4. **Utilizzare la gamification come leva di coinvolgimento**
   Incrementare l'engagement attraverso missioni quotidiane, sfide periodiche, premi virtuali e riconoscimenti sociali, per stimolare l'adozione continuativa e gratificante delle pratiche sostenibili.
5. **Favorire l'apprendimento esperienziale e partecipativo**
   Integrare elementi di verifica passiva (es. geolocalizzazione) e sfide collettive per rafforzare l'esperienza formativa attraverso l'interazione diretta e il confronto tra pari.
6. **Creare una comunità attiva e consapevole**
   Costruire una rete di utenti coinvolti, capaci di influenzare positivamente il contesto sociale e culturale di riferimento (scuole, famiglie, gruppi locali).

1.2. Deriverables

I deliverables avranno le scadenze indicate in seguito e saranno consegnati tramite **Moodle**:

- **D0**: Definizione dei gruppi di lavoro (7/10/2025)
- **D1**: Proposta iniziale (14/10/2025)
- **D2**: Piano di Progetto (21/10/2025)
- **D3**: Documento dei Requisiti (11/11/2025)
- **D4**: Piano di Testing (18/11/2025)
- **D5**: Documento di Progettazione (30/11/2025)
- **D6**: Versione 1.0 del codice sorgente (22/12/2025)
- **D7**: Versione 1.1. del codice e allineamento documentazione (26/01/2026)

Deliverable	Data pianificata	Data effettiva	Note
D0 - Gruppi di lavoro	07/10/2025	07/10/2025	In tempo
D1 - Proposta iniziale	14/10/2025	14/10/2025	In tempo
D2 - Piano di Progetto	21/10/2025	21/10/2025	In tempo
D3 - Documento Requisiti	11/11/2025	11/11/2025	In tempo
D4 - Piano di Testing	18/11/2025	18/11/2025	In tempo
D5 - Documento Progettazione	30/11/2025	30/11/2025	In tempo
D6 - Codice v1.0	22/12/2025	22/12/2025	In tempo
D7 - Codice v1.1 + allineamento docs	26/01/2026	25/01/2026	Consegna anticipata di 1 giorno

1.3. Evoluzione di progetto

L'app potrebbe distinguersi con un aspetto locale, oppure includere un wallet virtuale che converte i punti guadagnati in sconti o premi presso aziende partner.

Gli utenti possono guadagnare una "moneta virtuale" completando le quest, inoltre si possono perdere in caso le quest non vengano completate. Questa potrebbe essere utilizzata per una o più delle seguenti opzioni:

- Coltivare un orto virtuale acquistando piante e migliorandolo nel tempo.
- Partecipare a una classifica per rendere l'esperienza più competitiva.

È possibile creare una Green Routine personalizzata, con azioni come:

- andare in bici.
- fare docce brevi.
- spegnere le luci inutili.

Ogni settimana potrebbero essere proposte challenge speciali che offrono punti bonus e nel feed quotidiano si potrebbero trovare anche brevi "eco pill" con consigli e curiosità ecologiche.

2. Organizzazione del progetto

2.1. Modello del processo

Per lo sviluppo dell'applicazione, si è scelto di adottare un modello di sviluppo **Waterfall**, per i seguenti motivi:

- **Scaletta definita e milestone fisse:** Il progetto segue un calendario accademico con scadenze precise. Il modello Waterfall si adatta a questo tipo di contesto, consentendo una pianificazione rigorosa delle attività in fasi sequenziali.
- **Controllo e tracciabilità:** Ogni fase del progetto termina con la produzione di un deliverable e una revisione interna, consentendo un monitoraggio continuo dell'avanzamento. Questo è particolarmente utile in ambito accademico, dove sono richiesti report periodici e una documentazione coerente.
- **Team ridotto ma ben coordinato:** Nonostante il team sia composto da solo tre membri, la suddivisione chiara delle responsabilità e la linearità del processo Waterfall permettono una gestione efficace e una distribuzione equilibrata del lavoro.
- **Facilitazione della revisione e della valutazione:** L'approccio Waterfall consente di presentare lo stato di avanzamento in maniera ordinata e verificabile, favorendo il confronto con il docente nelle diverse fasi del progetto.

2.2. Struttura organizzativa

Il team adotta una **struttura decentralizzata**, coerente con la dimensione ridotta del gruppo, in cui le decisioni vengono prese in modo condiviso e i compiti sono distribuiti in base alle competenze e alla disponibilità. Non esiste una gerarchia fissa: tutti i membri partecipano attivamente alla pianificazione, allo sviluppo e alla verifica del prodotto.

2.3. Interfacce organizzative

Enrico Sforza svolge il ruolo di interfaccia principale con il docente responsabile del corso, è incaricato della consegna formale dei deliverable previsti, tra cui la versione preliminare dell'applicazione, la documentazione di accompagnamento e la versione finale, rispettando le scadenze stabilite dal calendario del corso.

2.4. Responsabilità del progetto

Alberto Ferragosti – Project Manager

Responsabilità principali:

- Pianificare le attività di progetto, definendo tempi, risorse e priorità.
- Coordinare il team e facilitare la comunicazione tra i membri.
- Monitorare l'avanzamento del progetto, garantendo il rispetto delle scadenze e degli obiettivi.
- Revisionare la documentazione tecnica e progettuale per assicurare coerenza e qualità.
- Gestire eventuali criticità, proponendo soluzioni e ricalibrando il piano operativo se necessario.

- Oltre alle responsabilità di pianificazione e coordinamento, il Project Manager ha svolto un contributo tecnico significativo sul backend (Node.js/Express, MongoDB, API design, autenticazione JWT e gestione sessione/token), compensando una sottostima iniziale dell'effort backend nella fase di sviluppo.

Enrico Rampazzo – Developer

Responsabilità principali:

- Sviluppare il codice secondo i requisiti funzionali e tecnici del progetto.
- Integrare servizi esterni tramite API e garantire il corretto funzionamento delle interfacce.
- Gestire e mantenere il repository GitHub, assicurando versionamento, ordine e tracciabilità delle modifiche.
- Collaborare con i designer e i tester per l'implementazione delle funzionalità richieste.

Enrico Sforza – Tester & UI Designer

Responsabilità principali:

- Progettare l'interfaccia utente in linea con le esigenze di usabilità e design.
- Eseguire test funzionali per verificare il corretto comportamento dell'applicazione.
- Validare la UI in termini di accessibilità, responsività e coerenza grafica.
- Redigere report di bug e segnalare anomalie al team di sviluppo per la loro risoluzione.

3. Processi gestionali

3.1. Obiettivi e priorità

Le scadenze del progetto sono **fisse**, definite in base alle tempistiche accademiche del corso.

In questo progetto si è deciso di **privilegiare la qualità** rispetto all'efficienza, in quanto l'obiettivo del corso non è soltanto realizzare un prodotto funzionante, ma anche **imparare** a progettare, sviluppare e validare soluzioni digitali in modo consapevole e professionale.

3.2. Assunzioni, dipendenze, vincoli

Assunzioni:

- Gli utenti hanno connessione Internet per interagire con il database.
- I dati ambientali (es. qualità dell'aria) sono forniti da servizi affidabili.

Dipendenze:

- Librerie Android (Google Maps API, Location Services).
- Versione minima Android 8 (API 26)

- Backend: Node.js + Express + Mongodb, hosting su render.com

- Sistema di autenticazione jwt

Vincoli:

- Tempo di sviluppo: fino a Gennaio.
- Team di 3 persone.
- Limite di budget (no API a pagamento).

3.3. Gestione dei rischi

ID	Rischio	Probabilità	Impatto	Mitigazione / Contromisure
1	Ritardo nello sviluppo di una funzionalità chiave (es. GPS o API qualità aria)	Media	Alta	Pianificare buffer di tempo, definire milestone chiare, assegnare responsabilità.
2	Problemi di performance o compatibilità su versioni Android più vecchie	Media	Media	Definire versione minima supportata, testare su vari dispositivi, ottimizzare codice.
3	Mancanza di coinvolgimento utenti/test (feedback scarso)	Media	Media	Prevedere fase di test con utenti, incentivare feedback, iterare rapidamente.
4	Disaccordi nel team / comunicazione inefficace	Media	Media	Stabilire riunioni regolari, usare strumenti condivisi (es. Git), definire processi.
5	Problemi di salute	Bassa-Media	Alta	Modifica alla divisione del lavoro

Oltre alla lista dei rischi e alle relative contromisure, viene introdotta una matrice Probabilità × Impatto per classificare rapidamente la criticità di ciascun rischio e definire le priorità di gestione.

Scala utilizzata

Probabilità (P)

1 = Bassa (evento poco probabile)

2 = Media (evento possibile)

3 = Alta (evento probabile)

Impatto (I)

1 = Basso (effetto limitato, recuperabile senza impatto su milestone)

2 = Medio (ritardo o degrado qualità gestibile con riorganizzazione)

3 = Alto (rischio di mancata consegna o compromissione requisiti principali)

Calcolo del Risk Score

Risk Score = P × I (valori possibili: 1–9)

Classificazione (soglie)

1–2 = Basso

3–4 = Medio

6–9 = Alto

Matrice Probabilità × Impatto (risk matrix)

PROBABILITA\IMPATTO	1 (basso)	2 (medio)	3 (alto)
Alta (3)	MEDIO	ALTO	ALTO
Media (3)	BASSO	MEDIO	ALTO
Bassa (1)	BASSO	BASSO	MEDIO

RISCHIO	PEOBABILITA	IMPATTO	SCORE	CLASSE
Ritardo nello sviluppo di funzionalità chiave (GPS / API)	2	3	6	ALTO
Performance/compatibilità su Android più vecch	2	2	4	MEDIO

Scarso coinvolgimento utenti/test (feedback insufficiente)	2	2	4	MEDIO
Disaccordi nel team / comunicazione inefficace	2	2	4	MEDIO
Problemi di salute / indisponibilità di un membro	2	3	6	ALTO

3.4. Meccanismi di controllo

La comunicazione avviene principalmente tramite un **gruppo WhatsApp**, usato per aggiornamenti rapidi e coordinamento quotidiano. Le decisioni importanti vengono riepilogate e riportate nella documentazione condivisa durante la revisione settimanale. Tutti i materiali di progetto (documenti, report) sono organizzati in **Google Drive**, accessibile a tutto il team.

3.5. Pianificazione dello staff

Enrico Sforza

Competenze tecniche

- Linguaggi: C, C++, Java
- Sistemi operativi: Windows, macOS
- Database: SQL (base)
- Reti: configurazioni base router/switch
- Hardware: assemblaggio, diagnostica, manutenzione
- Office: Word, Excel, PowerPoint

Competenze trasversali

- Lavoro di squadra
- Problem solving
- Gestione criticità

Lingue: Inglese B2

Alberto Ferragosti

Competenze tecniche

- Linguaggi: Python, JS, HTML/CSS, C++, PowerShell, AI
- Infrastruttura: VMware, Hyper-V, Windows Server, Linux, Active Directory
- Networking: LAN/WAN, VPN, VLAN, Firewall, Cisco, Unifi
- Cloud: Azure, AWS, Microsoft 365
- Sicurezza: Cybersecurity, GDPR, MFA, Backup, Disaster Recovery
- Data Science: Data Analysis, Machine Learning, Data Warehouse
- Project Management: Agile, Scrum, Waterfall, Risk Management

Competenze trasversali

- Leadership e gestione team
- Problem solving strategico
- Comunicazione tecnica
- Innovazione tecnologica

Lingue: Italiano (madrelingua), Inglese C1, Francese B2

Enrico Rampazzo

Competenze tecniche

- Linguaggi: C, C++, Java, HTML, CSS, PHP
- Sistemi operativi: Windows
- Database: SQL (base)
- Office: Word, Excel, PowerPoint

Competenze trasversali

- Problem solving
- Gestione criticità

Lingue: Inglese B2

Il team deve svolgere un corso di **programmazione Android**, visto che nessuno ha esperienza in questo ambito.

4. Processi tecnici

4.1. Metodi, strumenti e tecniche

L'ambiente di sviluppo utilizzato è **Android Studio**, IDE ufficiale fornito da Google per lo sviluppo di applicazioni Android.

Il front-end è sviluppato in **Java**, in quanto già conosciuto dal team, permettendo così di ridurre i tempi di apprendimento e concentrarsi maggiormente sulla realizzazione delle funzionalità.

Il backend è sviluppato in **JavaScript**

4.2. Documentazione

Ogni settimana il team si riunirà in **call** per effettuare una **revisione congiunta della documentazione**, verificando che i contenuti siano coerenti con l'avanzamento effettivo dello sviluppo, e per valutare l'eventuale necessità di modifiche o integrazioni.

Oltre alle revisioni settimanali, verranno effettuati aggiornamenti più sostanziali in corrispondenza delle due milestone principali: la **prima consegna** entro il 31 dicembre e la **versione finale** entro il 29 gennaio.

La **documentazione sarà conservata su Google Drive**, per facilitare la collaborazione e la modifica condivisa. Il codice verrà conservato nel **repository GitHub** del progetto.

5. Pianificazione delle attività e delle risorse

Il progetto seguirà un approccio **Waterfall**.

Ogni fase si conclude con la produzione di uno o più deliverable e una revisione interna per verificarne la completezza e la qualità. La durata complessiva del progetto è stimata da **ottobre 2025 a gennaio 2026**, in linea con le milestone accademiche.Il lavoro sarà distribuito in modo equilibrato tra i membri del team, garantendo collaborazione costante e controllo continuo dell'avanzamento.

5.1. WBS

La seguente tabella riassume la **scomposizione gerarchica delle attività principali (livello macro)**.

Ogni fase è associata a un deliverable specifico e rappresenta un passo logico del flusso di sviluppo.

Codice	Fase / Attività principale	Descrizione sintetica	Deliverable principale
1	**Analisi dei requisiti**	Raccolta, definizione e validazione dei requisiti funzionali e non funzionali. Identificazione delle macrofunzionalità.	Documento dei Requisiti
2	**Progettazione del sistema**	Definizione dell'architettura software, struttura dei moduli e interfacce principali (utenti, quest, gamification, impatto ambientale, eco-pillole, GPS).	Documento di Progettazione
3	**Progettazione UI/UX**	Ideazione dell'interfaccia utente, realizzazione di mockup e definizione dei flussi di navigazione.	Prototipo UI / Mockup grafici
4	**Sviluppo**	Implementazione del codice in Java mediante Android Studio. Integrazione delle API e delle funzionalità principali.	Codice sorgente
5	**Integrazione e testing**	Verifica funzionale, validazione delle prestazioni, correzione bug e ottimizzazione.	Piano di Testing , Versione 1.1

6	**Revisione e consegna finale**	Allineamento documentazione, revisione finale del codice e consegna su Moodle.	Versione finale e documentazione aggiornata

5.2. Dipendenze

Le attività del progetto presentano una sequenzialità logica, con alcune sovrapposizioni parziali tra progettazione e design UI.

La seguente tabella evidenzia le principali **dipendenze** tra le fasi:

Attività	Dipende da	Note
Progettazione del sistema	Analisi dei requisiti	Necessario completare la definizione funzionale prima dell'architettura
Progettazione UI/UX	Analisi e Progettazione	Può procedere in parallelo in alcune parti
Sviluppo	Progettazione sistema e UI	Richiede specifiche e mockup completati
Testing e Integrazione	Sviluppo	Inizia solo dopo la consegna del codice
Revisione finale	Testing	Include la validazione e la preparazione alla consegna

Schema di flusso sintetico:

Analisi → Progettazione → Sviluppo → Test → Consegna finale
↘ UI/UX ↗

5.3. Risorse necessarie

Le risorse impiegate nel progetto sono interamente interne al team e si basano su strumenti open-source o accademici.

Le attività vengono assegnate in base alle competenze e alle responsabilità dei membri.

Risorse umane

Nome	Ruolo	Responsabilità principali
Alberto Ferragosti	Project Manager	Pianificazione attività, coordinamento generale, revisione documentazione, monitoraggio avanzamento
Enrico Rampazzo	Developer	Implementazione codice, integrazione API, gestione repository GitHub
Enrico Sforza	Tester & UI Designer	Progettazione interfaccia, esecuzione test funzionali, validazione UI e report bug

Risorse tecniche

Categoria	Strumenti utilizzati
IDE e sviluppo	Android Studio
Linguaggio	Java
Versionamento codice	GitHub (repository privato)

Documentazione	Google Drive, Google Docs
Comunicazione interna	WhatsApp (aggiornamenti rapidi), Google Meet (riunioni periodiche)
Pianificazione e tracciamento	Foglio Google condiviso (Gantt semplificato)

5.4. Allocazione di budget e delle risorse

Il progetto non comporta costi economici diretti.

Le **risorse sono rappresentate dal tempo investito** dai membri del team, misurato in ore/uomo.

La tabella seguente mostra la stima per ciascuna fase:

5.1 Effort per fase: pianificato vs effettivo

Fase	Ore pianificate (D2)	Ore effettive	Scostamento	Nota
Analisi dei requisiti	**20h**	**18h**	**-2h (-10%)**	**Requisiti chiari fin dall'inizio**
Progettazione del sistema	**25h**	**20h**	**-5h (-20%)**	**Architettura semplificata (scope ridotto)**
Progettazione UI/UX	**15h**	**12h**	**-3h (-20%)**	**Riuso pattern Material Design**
Sviluppo	**60h**	**85h**	**+25h (+42%)**	**Backend/logiche quest sottostimate nella pianificazione**
Integrazione e testing	**25h**	**20h**	**-5h (-20%)**	**Test manuali vs automatici**
Revisione e consegna finale	**10h**	**8h**	**-2h (-20%)**	**Documentazione progressiva**

TOTALE	**155h**	**163h**	**+8h (+5%)**	**Scostamento complessivo contenuto**

5.2 Effort per membro (consuntivo)

Membro	Ruolo (D2)	Ore effettive	% effort	Attività prevalenti
Alberto Ferragosti	**Project Manager**	**60h**	**34%**	**Backend, DB, auth JWT, documentazione tecnica, coordinamento**
Enrico Rampazzo	**Developer**	**58h**	**33%**	**Frontend Android, quest system, adapters, UI quest**
Enrico Sforza	**Tester & UI Designer**	**58h**	**33%**	**UI/UX, testing, backend social (friends/badges)**

5.5. Diagramma di Gantt

5.5. Diagramma di Gantt

Il diagramma di Gantt rappresenta la pianificazione temporale delle attività principali del progetto nel periodo ottobre 2025 – gennaio 2026.

Il Gantt è stato derivato dal diagramma di PERT (sezione 5.6), tenendo conto delle dipendenze tra attività e del percorso critico individuato.

Per aumentare la tracciabilità, oltre alle macro-attività sono indicate anche alcune sotto-attività interne (secondo livello) che aiutano a controllare l'avanzamento e a ridurre il rischio di ritardi concentrati nella fase di sviluppo.

Assunzione temporale di riferimento

1 settimana = 7 giorni di calendario (coerente con PERT).

Le date indicate sono stime e possono subire piccoli aggiustamenti.

Attività e sotto-attività (secondo livello) per controllo avanzamento

- A) Analisi dei requisiti (1 settimana)
    - A1 Raccolta requisiti e casi d'uso
    - A2 Revisione e validazione requisiti (baseline per D3)
- B) Progettazione del sistema (1,5 settimane)
    - B1 Architettura (client Android + backend Node/Express + DB MongoDB)
    - B2 Modellazione dati e API (endpoints principali)
    - B3 Bozza progettazione sicurezza (JWT, policy privacy) (baseline per D5)
- C) Progettazione UI/UX (1 settimana, parzialmente sovrapposta)
    - C1 User flow principali (login/registrazione, missioni, profilo)
    - C2 Mockup schermate e stile (baseline mockup UI)
- D) Sviluppo (4 settimane)
    - D1 Setup progetto (repo, struttura app, build, dipendenze)
    - D2 Implementazione autenticazione (login/register, token manager)
    - D3 Implementazione missioni/quest e gamification (punti, badge, level)
    - D4 Implementazione dashboard impatto (CO2, progressi)
    - D5 Integrazione GPS/geolocalizzazione (verifica passiva)
    - D6 Integrazione backend (API, DB, servizi) (baseline per D6)
- E) Integrazione e testing (2 settimane)
    - E1 Test funzionali principali (happy path e casi di errore)
    - E2 Test compatibilità (Android min 8 / API 26) e performance base
    - E3 Bug fixing e stabilizzazione (baseline D4 e D7)
- F) Revisione e consegna finale (1 settimana)
    - F1 Allineamento documentazione e codice
    - F2 Pacchetto consegna e verifica finale

Indicazione del percorso critico (derivato dal PERT)

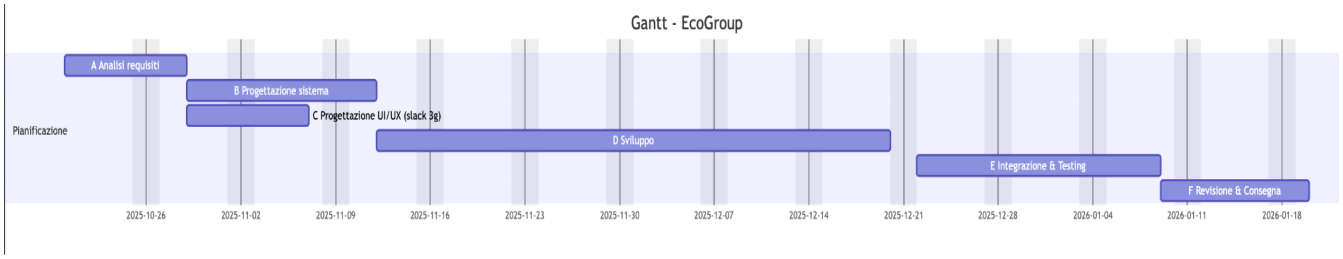
Il percorso critico include le attività A → B → D → E → F.

L'attività C (UI/UX) può sovrapporsi parzialmente a B e presenta slack: eventuali piccoli ritardi su C non impattano direttamente la data finale, purché non superino il margine disponibile.

Cronologia sintetica (ottobre 2025 – gennaio 2026)

- Settimana 1 (Ott): Analisi requisiti (A)
- Settimane 2–3 (Ott/inizio Nov): Progettazione sistema (B) e avvio UI/UX (C in parallelo)
- Settimane 4–7 (Nov–Dic): Sviluppo (D)
- Settimane 8–9 (Dic–Gen): Integrazione e testing (E)
- Settimana 10 (Gen): Revisione e consegna finale (F)

Nota: il diagramma grafico allegato (Gantt) evidenzia le sovrapposizioni tra B e C e la collocazione delle macro-fasi nel calendario.



5.6. Diagramma di PERT

Il diagramma PERT descrive le dipendenze tra le attività e consente di individuare il percorso critico attraverso il calcolo dei tempi al più presto (ES/EF) e al più tardi (LS/LF), oltre allo slack (margine).

Per rendere il PERT significativo, la fase UI/UX è stata modellata come attività eseguibile in parallelo rispetto alla progettazione del sistema dopo la conclusione dell'analisi, coerentemente con la possibilità di iniziare i mockup una volta stabiliti i requisiti.

Attività, durate e predecessori

ID	ATTIVITA	DURATA (GIORNI)	PREDECESSORI
A	Analisi dei requisiti	7	-
B	Progettazione del Sistema	10	A
C	Progettazione UI/UX	7	A
D	Sviluppo	28	B, C
E	Integrazione e Testing	14	D
F	Revisione e consegna finale	7	E

Calcolo tempi al più presto (Forward pass: ES/EF)

- A: ES=0 EF=0+7 =7
- B: ES=7 EF=7+10 =17
- C: ES=7 EF=7+7 =14
- D: ES=max(EF(B),EF(C))=max(17,14)=17 EF=17+28=45
- E: ES=45 EF=45+14=59
- F: ES=59 EF=59+7 =66

Durata totale del progetto (stimata): 66 giorni

Calcolo tempi al più tardi (Backward pass: LS/LF)

- F: LF=66 LS=66-7 =59
- E: LF=59 LS=59-14=45
- D: LF=45 LS=45-28=17
- B: LF=LS(D)=17 LS=17-10=7
- C: LF=LS(D)=17 LS=17-7 =10
- A: LF=min(LS(B),LS(C))=min(7,10)=7 LS=7-7=0

Tabella riassuntiva ES/EF/LS/LF e Slack

ID	DURATA	ES	EF	LS	LF	SLACK (LSES)
A	7	0	7	0	7	0
B	10	7	17	7	17	0
C	7	7	14	10	17	3
D	28	17	45	17	45	0
E	14	45	59	45	59	0
F	7	59	66	59	66	0

Percorso critico

Le attività con Slack = 0 costituiscono il percorso critico:

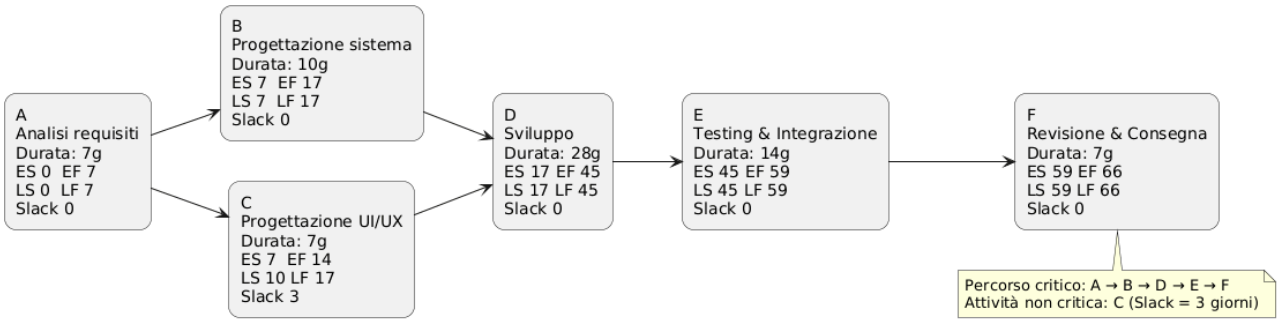
A → B → D → E → F

L'attività C (UI/UX) non è critica e presenta uno Slack pari a 3 giorni: ciò significa che può ritardare fino a 3 giorni senza impattare la data finale, purché sia completata entro LF=17 (prima dell'avvio di D).

Implicazioni operative

Eventuali ritardi nelle attività A, B, D, E o F causano uno slittamento diretto della consegna finale.

Ritardi limitati dell'attività C sono assorbibili entro il margine (slack), ma oltre tale soglia bloccano l'avvio dello sviluppo (D).



Considerazioni

- Le attività C (UI/UX) e B (Progettazione sistema) possono avere parziale sovrapposizione iniziale.
- Il percorso critico attraversa tutte le attività principali, poiché il completamento di ciascuna è prerequisito per la successiva.
- Un eventuale ritardo nello Sviluppo (D) o nel Testing (E) impatterebbe direttamente sulla consegna finale.
- L'attività F include anche la revisione documentale e la presentazione.