Progetto di Ingegneria del Software 2023/24
Università Ca' Foscari Venezia

Piano di progetto

Versione 1.5

GrindTeam

17/10/2023

Document Informations

FillNDrive	FND
Deliverable	Piano di progetto
Data di Consegna	17/10/2023
Team Leader	Beatrice Gigli 869968@stud.unive.it
Team members	Enrico Giacobbi 890428@stud.unive.it, Federico Nori 890184@stud.unive.it, Niccolò Pirillo 890361@stud.unive.it, Riccardo Vendraminetto 892512@stud.unive.it

Document History

Version	Issue Date	Stage	Changes	Contributors
1.0	03/10/2023	Draft	Stesura dello scheletro del documento.	Beatrice Gigli
1.1	09/10/2023	Draft	Compilato paragrafo 4.	Beatrice Gigli
1.2	14/10/2023	Draft	Compilati paragrafi 1, 2, 3.	Enrico Giacobbi
1.3	14/10/2023	Draft	Compilato paragrafo 5.	Niccolò Pirillo
1.4	16/10/2023	Draft	Inserito sottoparagrafo 1.2 Overview del progetto; Inserita seconda parte del paragrafo 2.1 Modello del Processo; Revisionato il contenuto dell'intero documento e applicata conformità nel formato del testo.	Beatrice Gigli
1.5	17/10/2023	Final	Ultime modifiche e revisione finale	Niccolò Pirillo, Beatrice Gigli

Indice

1. Introduzione

1.1. Executive Summary

Il documento fornisce una guida dettagliata su come il progetto sarà pianificato, eseguito, monitorato e controllato. Contiene informazioni essenziali come obiettivi, scadenze, risorse, budget, responsabilità dei membri del team e procedure per gestire eventuali deviazioni dal piano originale. Serve come strumento fondamentale per il coordinamento e il successo del progetto.

1.2. Overview del Progetto

Il progetto consiste in un'applicazione android che permette all'utente, una volta scelte alcune impostazioni personalizzate, di visualizzare una mappa contenente le stazioni di rifornimento più vicine alla sua posizione con lo scopo di agevolare il conducente nella scelta della stazione più idonea per il rifornimento del proprio veicolo.
Ciascuna stazione viene indicata con un marker nella mappa, corredato da un'etichetta che indica il prezzo del carburante offerto. I marker assumono colore rosso, giallo oppure verde in base al prezzo che la stazione espone rispetto al prezzo medio provinciale.

L'utente può zoomare la mappa in avanti o indietro, cambiando l'ampiezza di visibilità nel territorio in modo da considerare un numero differente di stazioni di rifornimento; può inoltre cliccare i marker per avviare un itinerario di navigazione verso una specifica stazione.

Dopo un'attenta analisi di mercato riguardo le applicazioni già esistenti e con lo scopo di offrire un servizio non ancora fruibile da Google Play Store, il team ha stabilito una revisione della proposta iniziale (deliverable D0) mediante l'implementazione di un'unica mappa nell'applicazione. Tale mappa indica all'utente la stazione di rifornimento per lui più conveniente, scelta da un algoritmo che considera il rapporto tra la distanza e il prezzo. In questo modo, l'applicazione suggerirà al conducente la stazione di servizio con un prezzo inferiore, che sia anche sufficientemente vicina in modo da minimizzare il consumo di carburante necessario per raggiungerla, garantendo così che il risparmio rispetto ad altre stazioni di rifornimento sia ottimizzato.

L'utente è altresì libero di ignorare la stazione di servizio consigliata dall'algoritmo e scegliere autonomamente verso quale stazione avviare la navigazione, in quanto le altre stazioni presenti nel raggio chilometrico selezionato saranno comunque visibili sulla mappa, seppur contrassegnate da marker di dimensioni inferiori.

1.3. Deliverables del progetto

Il progetto dovrà essere corredato da alcune deliverables con le seguenti scadenze:

- D0: Proposta Iniziale, fissata per il 3 ottobre 2023
- D1: Piano di Progetto, previsto per il 17 ottobre 2023
- D2: Documento dei Requisiti, stabilito per il 31 ottobre 2023
- D3: Piano di Testing, fissato per il 14 novembre 2023
- D4: Documento di Progettazione, previsto per il 28 novembre 2023
- D5: Versione 1.0 del Codice Sorgente, stabilita per il 15 dicembre 2023

- D6: Versione 1.1 del Codice e Allineamento Documentazione, fissato per il 15 gennaio 2024

1.4. Evoluzione del progetto

Attualmente il piano di lavoro prevede che i marker vengano colorati sulla base del prezzo medio provinciale del carburante. Come evolutiva futura, si potrebbe modificare tale criterio colorandoli invece in base al *risparmio* medio, valutato considerando tutte le stazioni di rifornimento presenti nella mappa.
In questo modo, l'indicazione del colore permetterà all'utente di capire se una stazione di servizio è più o meno conveniente delle altre presenti nel raggio chilometrico impostato.
Si ricorda che il risparmio viene calcolato sulla base del prezzo del carburante offerto da una particolare stazione di rifornimento e della quantità di carburante che il conducente consuma per raggiungerla.

Imposteremo l'applicazione in modo che sia facilmente flessibile ai cambiamenti o a delle aggiunte, ad esempio si potrebbe pensare di aggiungere le stazioni di servizio che offrono le colonnine di ricarica per i veicoli elettrici.
Si potrebbe inoltre predisporre l'applicazione alla gestione di altri luoghi di interesse oltre alle stazioni di rifornimento, come parcheggi, piazzole di sosta, ecc.

1.5. Materiali di riferimento

- Slide del corso di Ingegneria del Software
- StackOverflow
- Documentazione Google per Android
- Android studio
- Documentazione SDK

1.6. Definizioni e abbreviazioni

Android è un sistema operativo per dispositivi mobili sviluppato da Google, progettato principalmente per sistemi embedded quali smartphone e tablet.

Android Studio è un ambiente di sviluppo integrato (IDE) per lo sviluppo per la piattaforma Android.

App In informatica, un'applicazione mobile è un'applicazione software dedicata ai dispositivi di tipo mobile, quali smartphone o tablet. Con questo termine d'ora in poi ci si riferisce all'applicazione Android del progetto che il team svilupperà.

GitHub è un servizio di hosting per progetti software, e permette tramite l'integrazione di Git di tenere traccia delle diverse versioni del codice sorgente e di file di diversa natura.

Trello è un software gestionale in stile Kanban basato sul web.

2. Organizzazione del progetto

2.1. Modello del processo

Per la realizzazione del progetto, il team ha adottato il modello **Agile**.

Vantaggi:

- **Adattabilità ai cambiamenti**: consente di apportare modifiche ai requisiti, alle funzionalità e alle priorità del progetto in qualsiasi momento;
- **Migliore gestione del rischio**: l'approccio iterativo permette di identificare e affrontare i problemi in modo tempestivo;
- **Maggiore coinvolgimento del team**;
- **Maggiore qualità del prodotto**: l'attenzione costante alla qualità è un principio fondamentale dell'Agile;
- **Apprendimento continuo**: l'approccio Agile promuove l'apprendimento continuo e l'evoluzione del processo;
- **Maggiore controllo del budget.**

Svantaggi:

- **Complessità della gestione;**
- **Dipendenza dalla comunicazione;**
- **Scalabilità.**

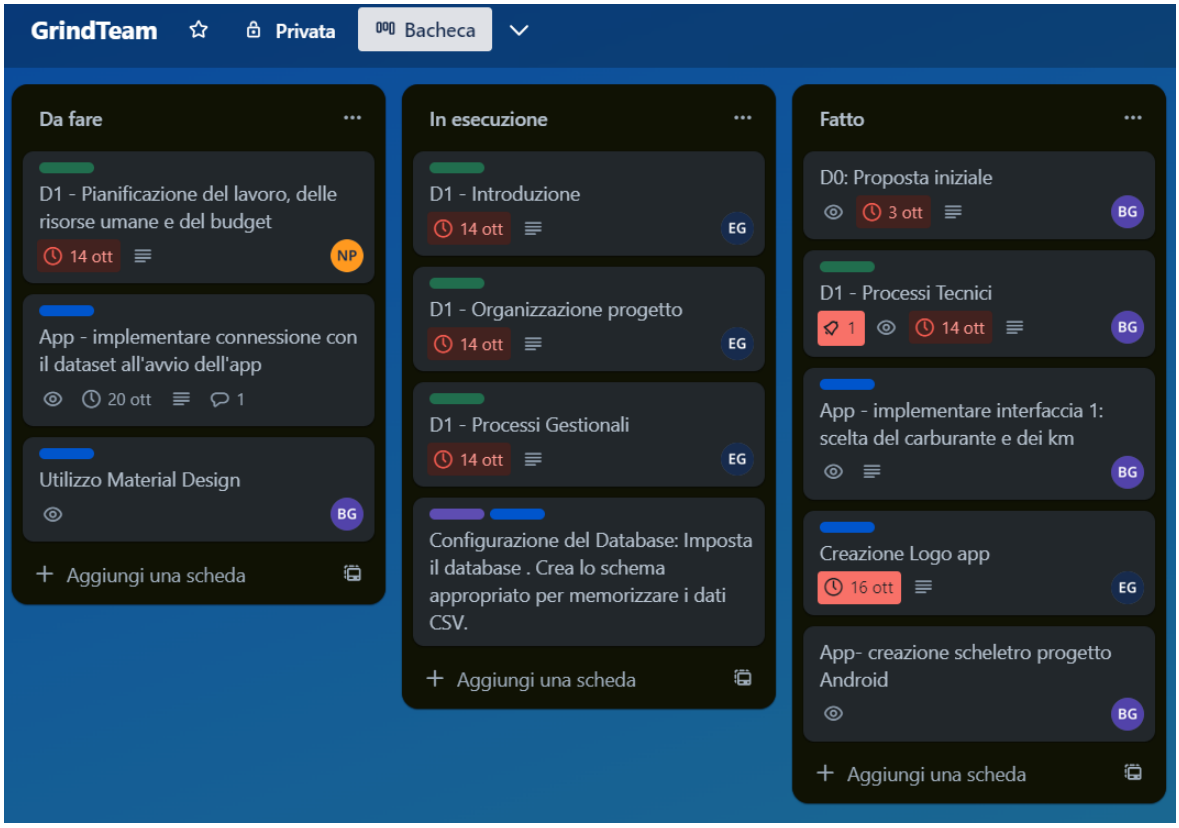
Il progetto viene suddiviso in più fasi, dette *sprint*, in riferimento alla metodologia **Agile Scrum**.
Ogni sprint è composto da diversi *task*, ovvero piccole attività da prendere in carico che coesistono all'interno del medesimo sprint e che, talvolta, dipendono l'una dall'altra.
Al fine di garantire una distribuzione equa del carico di lavoro tra il personale, ciascun membro del team prenderà in carico alcuni task da portare a termine entro la fine dello sprint. Solitamente uno sprint ha durata di una o due settimane al massimo.

Risulta quindi di fondamentale importanza la coordinazione tra i membri del team: a questo proposito, si è deciso di adottare **Trello**.
Si tratta di una board condivisa contenente alcune card, una per ogni task da completare. Ogni card è caratterizzata da una descrizione dettagliata dell'attività che deve essere svolta, dalla data di scadenza - che in genere coincide con la fine dello sprint -, e il nome della persona responsabile dell'attività.

Inoltre è possibile aggiungere dei commenti alle card in modo da avere uno storico degli sviluppi che l'attività richiede e tenere traccia di eventuali problematiche emerse durante lo svolgimento dell'attività.

In questo contesto ogni componente del team ha la responsabilità di tenere aggiornata la board in relazione alle proprie card, inserendo commenti rilevanti e spostando le card nelle colonne adeguate man mano che le attività avanzano. Solo in questo modo si può garantire un'efficace coordinazione del team di lavoro, consentendo a tutti di avere sotto controllo un'immagine precisa dello stato di avanzamento del progetto.

Di seguito viene riportato un estratto della bacheca di cui il team si è servito per questo sprint.



2.2. Struttura organizzativa

Il team adotta una struttura di tipo democratico decentralizzato. Questo implica un processo decisionale collettivo attraverso incontri periodici, anche in modalità virtuale utilizzando la piattaforma Google Meet, quando le riunioni in presenza non sono praticabili. Con la progressione dello sviluppo, l'allocazione dei compiti verrà effettuata in base alle competenze specifiche dei membri del team, con l'obiettivo di garantire la produzione di un prodotto finale di qualità.

Nel team è presente una figura di riferimento che svolge il ruolo di team leader, che si occupa dell'organizzazione del lavoro e del coordinamento del personale.

2.3. Interfacce organizzative

Durante lo svolgimento del progetto ci sarà la necessità di comunicare con entità esterne al gruppo. A questo proposito, il team leader avrà il compito di effettuare consegne periodiche delle deliverables entro le scadenze prefissate.
Inoltre tutti i membri del gruppo avranno il compito di seguire attentamente le lezioni e nel caso di dubbi porre domande sul tema organizzativo al Professor A. Cortesi e domande di tipo tecnico al Professor A. Spanò.

Sarà necessario interfacciarsi con i futuri tester per avere feedback da loro e applicare modifiche in base ai problemi sorti.

2.4. Responsabilità di Progetto

Le decisioni dei task si prendono di comune accordo tra tutti i componenti del team.

Ogni task verrà assegnato in modo equo al personale in modo da non sovraccaricare singoli membri del team; a tal proposito, possono essere creati occasionalmente dei sottogruppi formati da due o tre persone per la gestione dei task più complessi.

Ogni componente del team, oltre allo svolgimento dei propri task, partecipa a momenti di condivisione dello stato di avanzamento delle varie attività in carico agli altri membri, in modo da avere una visione complessiva dello stato attuale del progetto e poter offrire il proprio contributo scambiando opinioni e consigli.

3. Processi Gestionali

3.1. Obiettivi e Priorità

Il gruppo si pone come priorità e obiettivi:

- Produrre un prodotto di qualità;
- Rispettare le scadenze stabilite;
- Collaborare, e mantenere coesione;
- Rispettare i vincoli del progetto.

3.2. Assunzioni, Dipendenze, Vincoli

Assunzioni: Assumiamo che dopo le lezioni di Android (presentate dal Prof. Spanò) tutti i membri del gruppo abbiano una buona comprensione dello sviluppo su Android Studio, anche se saranno necessari sicuramente successivi approfondimenti, inoltre si suppone che tutti abbiano una buona familiarità nella costruzione di una base di dati. Ci si aspetta da ogni membro del gruppo una cospicua partecipazione per portare a termine il lavoro.

Dipendenze: Per poter cominciare a sviluppare la nostra applicazione dovremmo utilizzare una base di dati, di conseguenza sarà necessario crearla e mantenerla. Inoltre, per realizzare una mappa interattiva sarà richiesto l'utilizzo delle API Google Maps.

Vincoli: il rispetto delle consegne prestabilite dal professore, la realizzazione di un'applicazione completa, affidabile e che offra una buona user experience.

3.3. Gestione dei rischi

Per un'efficace pianificazione delle attività, è fondamentale identificare gli elementi che possono pregiudicare il successo del progetto.

Essendo un progetto che si sviluppa in un intervallo di tempo prestabilito, è necessario prendere in considerazione eventuali rischi che possono insorgere e fare in modo che vengano gestiti lavorando in gruppo. Di seguito se ne riportano alcuni.

Bugs: in fase di sviluppo è molto probabile individuare errori nel codice. Per evitare situazioni spiacevoli e inutili perdite di tempo, è necessario definire un appropriato piano di testing che consenta di trovare bug in modo da procedere con l'immediata correzione. Inoltre si potrebbe attuare il peer-programming per i task che verranno eseguiti da piccoli sottogruppi di persone. Questo tipo di rischio ha un impatto variabile, in quanto un bug potrebbe essere subito scovato e corretto facilmente in modo efficace, oppure potrebbe risultare complesso da individuare e gestire.

Difficoltà con il linguaggio di programmazione: i membri del gruppo si devono impegnare a colmare le lacune relative alla conoscenza del linguaggio di programmazione (se presenti) e collaborare per risolvere i relativi problemi nel caso il singolo membro non riesca a trovare una soluzione.

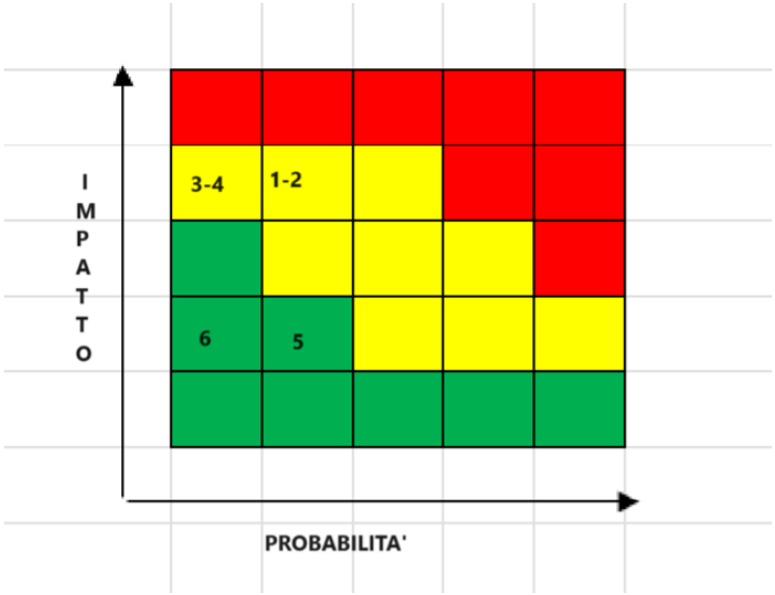
Guasto hardware ai computer: se dovesse avvenire un guasto hardware ad una macchina da noi utilizzata per sviluppare il progetto, la soluzione è quella di utilizzare momentaneamente un'altra macchina per ovviare a questo problema, e successivamente risolvere il problema al pc guasto.

Guasto software ai computer: questo rischio di per sé non è un gran problema, perché con le nostre conoscenze siamo in grado di riparare il danno. Se dovesse avvenire un guasto software, il problema più grande sarebbe la perdita di tempo per rimediare al guasto. Il caso peggiore prevede una formattazione del disco e una reinstallazione dei software utilizzati per lo sviluppo dell'applicazione.

Perdita del lavoro fatto: questo rischio potrebbe danneggiare irrimediabilmente il progetto, è necessario mantenere più versioni di backup mediante google drive e github.

Problemi di salute: in questo periodo c'è un maggior rischio di ammalarsi e di conseguenza rallentare lo sviluppo del progetto. In questo caso bisogna essere pronti a rimediare all'inconveniente aiutandosi a vicenda senza lasciare che il proseguimento dei lavori ne risenta.

	Tabella dei rischi	**Probabilità**	**Impatto**
1	Non finire in tempo il progetto	2	4
2	Non rispettare le scadenze delle singole attività	2	4
3	Divergenza fra componenti del gruppo	1	3
4	Sottostima della difficoltà del progetto	1	3
5	App poco attraente per gli utenti	2	2
6	Variazione formato dei dati	1	2

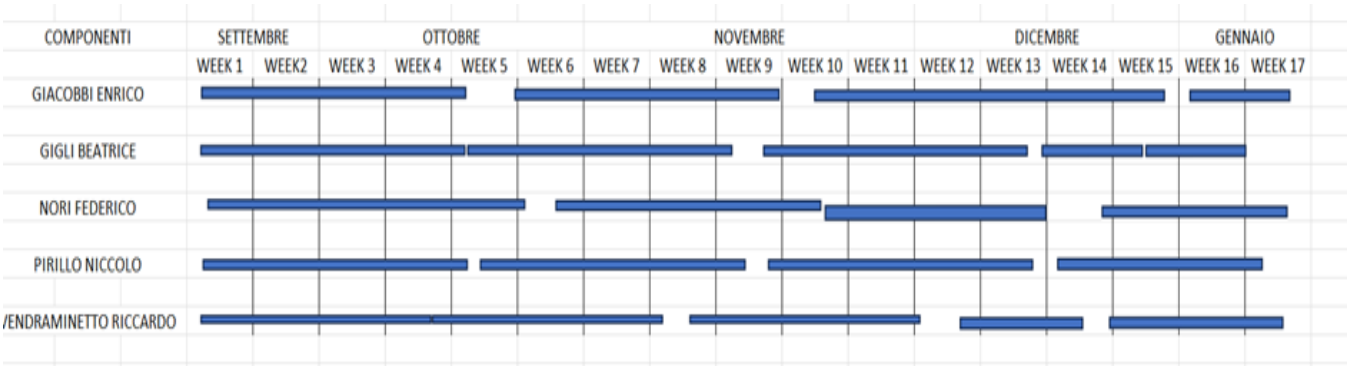


3.4. Meccanismi di monitoraggio e di controllo

Il progetto verrà sviluppato in gruppo, di conseguenza il tutto sarà sotto l'occhio vigile dei componenti del team in modo da avere un controllo incrociato su documentazione e codice sorgente. Talvolta i task complessi potranno essere assegnati a sottogruppi di poche persone che lavoreranno in maniera trasversale, in modo da effettuare un controllo più intenso nelle attività critiche, oltre che velocizzare il lavoro.

Per potersi confrontare si utilizzeranno i canali google meet e WhatsApp per tenere in comunicazione i membri del gruppo, Google Documents per la revisione della documentazione e GitHub per il codice sorgente.

3.5. Pianificazione dello staff



4. Processi Tecnici

4.1. Metodi, Strumenti e Tecniche

Sistemi di calcolo

Per lo sviluppo del progetto verrà utilizzata l'ultima versione dell'IDE Android Studio, che ad oggi è **Android Studio Giraffe 2022.3.1**, disponibile per i principali sistemi operativi (Microsoft Windows, Linux, MacOS, ChromeOS).
Una delle caratteristiche distintive delle applicazioni mobile per Android è la loro indipendenza dal sistema operativo del calcolatore utilizzato per lo sviluppo del codice. In questo contesto, ciascun membro del team può scegliere in autonomia il sistema operativo su cui installare e configurare l'IDE.
Un altro vantaggio di Android Studio è la presenza dell'SDK - Software Development Kit -, che mette a disposizione un emulatore di dispositivi Android. L'emulatore è un dispositivo virtuale che consente di testare le applicazioni sviluppate senza la necessità di disporre di un dispositivo fisico apposito.
L'attività di debug e testing dell'applicazione può essere svolta sia con il supporto dell'emulatore, sia con il dispositivo Android posseduto da ogni membro. È importante sottolineare che la versione minima richiesta per poter installare ed eseguire l'applicazione è **Android 7.0** o superiore.

Metodi di Sviluppo

Ogni membro del team collabora nella stesura delle documentazioni e nello sviluppo dell'applicazione.
Per coordinare efficacemente il lavoro di gruppo, verranno utilizzate le seguenti tecnologie:

- **GitHub**: repository remoto per il versionamento del codice sorgente dell'applicazione. Nel repository verranno caricate anche le varie versioni delle deliverables;
- **Google Docs**: per redarre le deliverables in modo che ogni persona possa modificare il documento in maniera simultanea e condivisa con gli altri membri. Ciascuna versione dello stesso documento verrà successivamente caricata nel repository GitHub in modo da mantenere lo storico della documentazione;
- **Whatsapp**: per la comunicazione diretta tra i membri del team;
- **Trello**: bacheca virtuale condivisa tra tutti i membri per la suddivisione delle attività;

Strumenti

- **Excel** per la realizzazione del diagramma di Gantt;
- **One Note**, per la realizzazione del diagramma di Pert;
- **Material Design** per la realizzazione delle interfacce grafiche;
- **Maps SDK for Android** per l'implementazione della mappa.

Da scegliere, dopo attenta analisi tecnica, l'utilizzo di FireBase oppure SQLite come database su cui l'applicazione si appoggerà per la lettura del dataset.

Tecniche

L'applicazione sarà sviluppata a partire da una base predefinita fornita in bundle con Android Studio, che predispone una semplice navigazione tra diverse interfacce utente. Questa verrà poi estesa con l'implementazione di tutte le funzionalità che il progetto si propone di fornire: dalla comunicazione con il

database e lo scarico dei dati in tempo reale, fino all'implementazione delle interfacce grafiche, che includeranno la localizzazione del dispositivo, le mappe e i filtri per l'utente.
Ci si affida alle linee guida disponibili nel sito ufficiale developer.android.com e alle Java best practices.

4.2. Documentazione del software

Piano di documentazione: documenti in .doc condivisi nel repository Git.

È prevista la documentazione del software sviluppato, che si articola nei seguenti documenti:

- Piano di Progetto
- Documento dei Requisiti
- Piano di Testing
- Documento di Progettazione

Si faccia riferimento al rif. 1.3 per le date di consegna.
Le documentazioni saranno fornite in formato PDF seguendo la struttura definita nel file Format_Progetto_23 di seguito riportato.

Format_Progetto_23.pdf

4.3. Funzionalità di supporto al progetto

Per garantire un'efficace realizzazione del progetto, il team si atterrà alle linee guida organizzative e ai requisiti del progetto stabiliti in questa fase, evitando continui cambiamente alla struttura esterna dell'applicazione.
Verranno seguite le euristiche di programmazione e testate di volta in volta le macro-implementazioni apportate al progetto al fine di valutarne la consistenza e rilevare eventuali bug nel software.

Particolare attenzione verrà dedicata alla parte documentativa, che sarà costantemente aggiornata con le nuove evidenze che emergeranno soprattutto durante la fase iniziale di avvio del progetto. A tendere, tali evidenze saranno consolidate in strategie condivise da tutto il team.

Per coordinare i vari membri del team e permettere il lavoro simultaneo sulle risorse condivise, il team si avvale delle tecnologie sopra citate [rif. 4.1]. In particolare, si utilizzerà Trello per la gestione del personale, GoogleDocs per la collaborazione nella creazione dei documenti, e GitHub per il versionamento del codice sorgente e della documentazione.

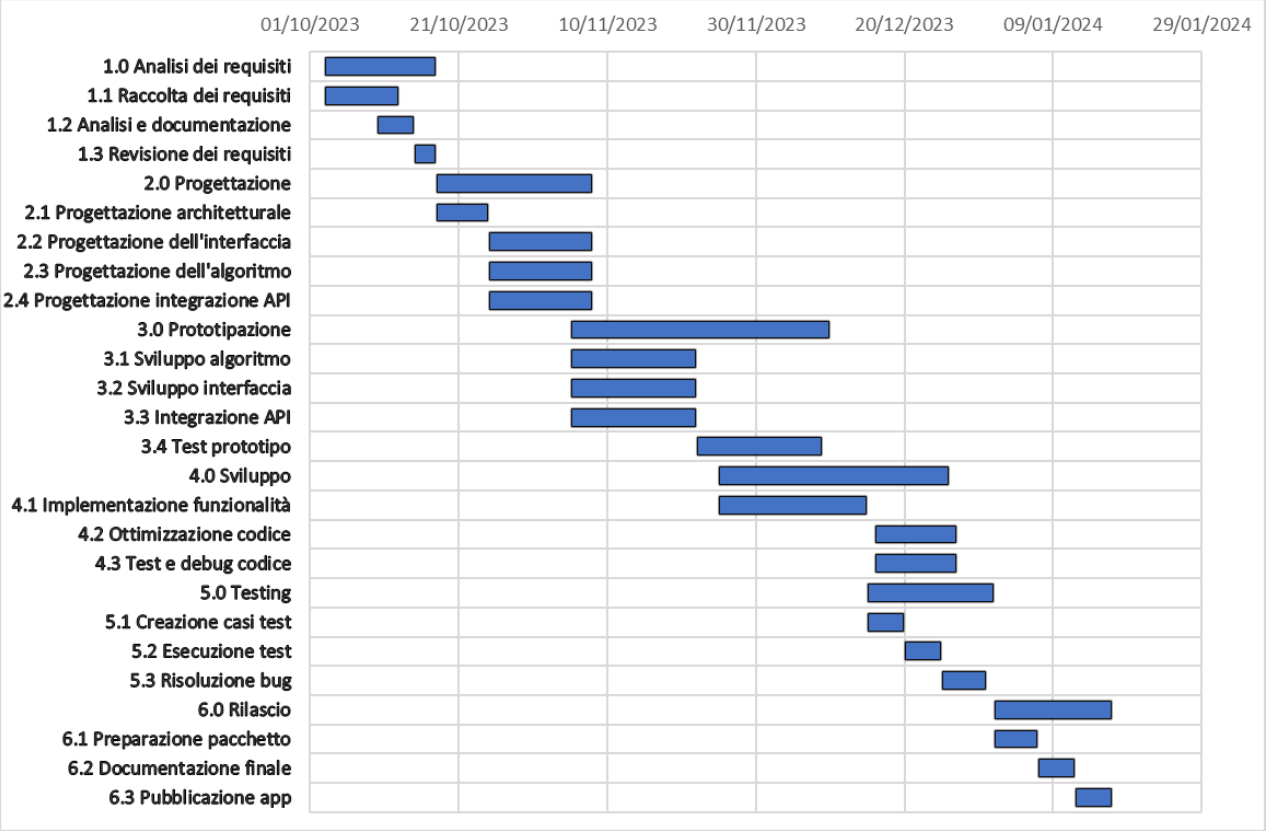
5. Pianificazione del lavoro, delle risorse umane e del budget

5.1. Work Breakdown Structure

Il progetto sarà suddiviso nelle seguenti attività:

- **1.0 Analisi dei requisiti**
- 1.1 Raccolta dei requisiti
- 1.2 Analisi e documentazione
- 1.3 Revisione dei requisiti
- **2.0 Progettazione**
- 2.1 Progettazione architetturale
- 2.2 Progettazione dell'interfaccia
- 2.3 Progettazione dell'algoritmo
- 2.4 Progettazione integrazione API
- **3.0 Prototipazione**
- 3.1 Sviluppo algoritmo
- 3.2 Sviluppo interfaccia
- 3.3 Integrazione API
- 3.4 Test prototipo
- **4.0 Sviluppo**
- 4.1 Implementazione funzionalità
- 4.2 Ottimizzazione codice
- 4.3 Test e debug codice
- **5.0 Testing**
- 5.1 Creazione casi test
- 5.2 Esecuzione test
- 5.3 Risoluzione bug
- **6.0 Rilascio**
- 6.1 Preparazione pacchetto
- 6.2 Documentazione finale
- 6.3 Pubblicazione app

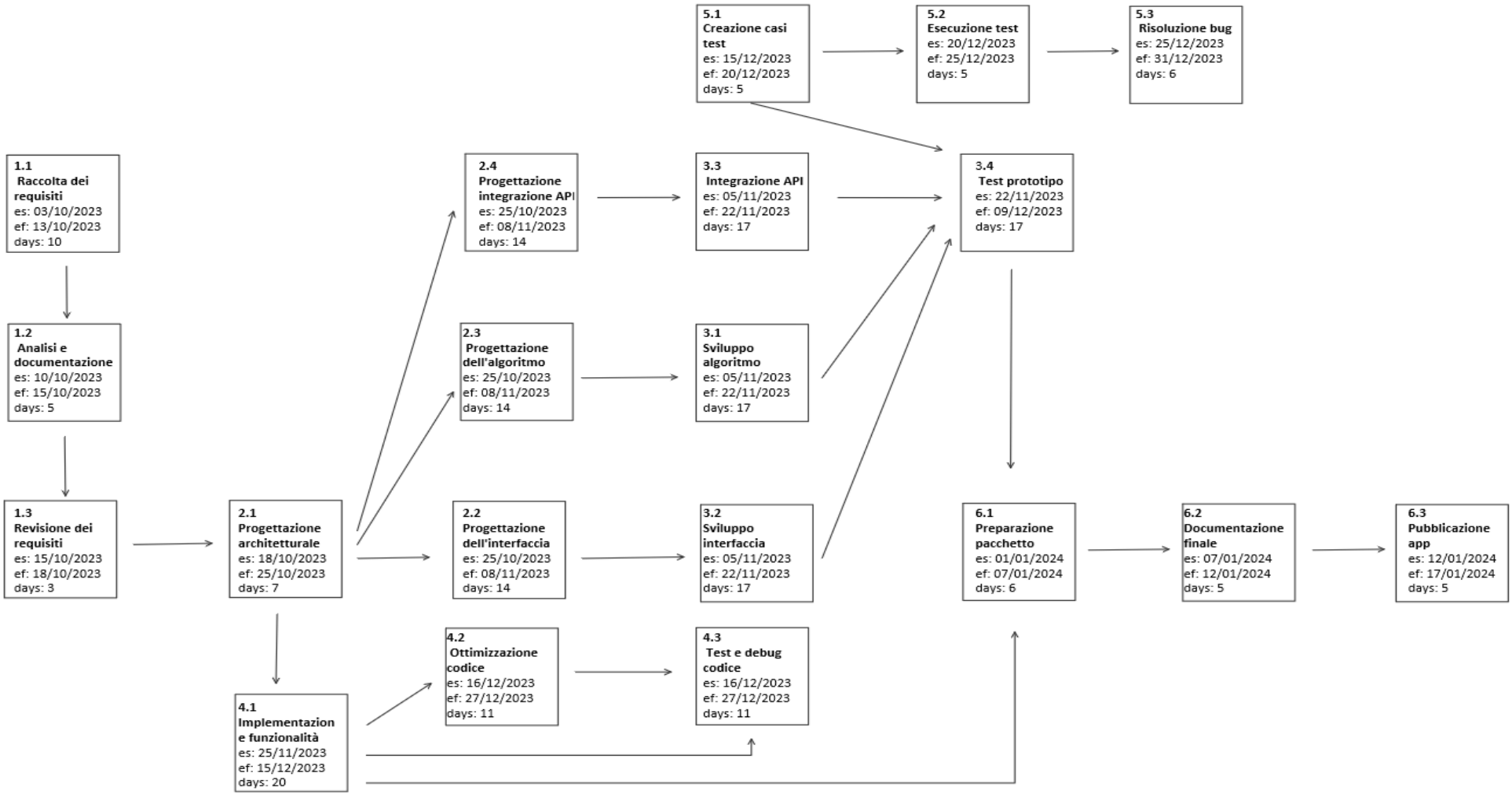
5.1.1. Diagramma di Gnatt



5.1.2. Diagramma di Pert

Si faccia riferimento alla seguente legenda per interpretare il diagramma di Pert.

- **ES** (earliest start time): indica il minimo giorno di inizio dell'attività, calcolato sulla base del minimo tempo necessario per terminare le attività che precedono quella corrente;
- **EF** (earliest finish time): noti il tempo ES e la durata dell'attività, indica il minimo giorno in cui l'attività può terminare;
- **Durata**: indica quanti giorni di lavoro sono necessari per prendere in carico e terminare l'attività.


1.1
Raccolta dei requisiti
es: 03/10/2023
ef: 13/10/2023
days: 10
1.2
Analisi e documentazione
es: 10/10/2023
ef: 15/10/2023
days: 5
1.3
Revisione dei requisiti
es: 15/10/2023
ef: 18/10/2023
days: 3
2.1
Progettazione architetturale
es: 18/10/2023
ef: 25/10/2023
days: 7
2.4
Progettazione integrazione API
es: 25/10/2023
ef: 08/11/2023
days: 14
2.3
Progettazione dell'algoritmo
es: 25/10/2023
ef: 08/11/2023
days: 14
2.2
Progettazione dell'interfaccia
es: 25/10/2023
ef: 08/11/2023
days: 14
4.1
Implementazione funzionalità
es: 25/11/2023
ef: 15/12/2023
days: 20
4.2
Ottimizzazione codice
es: 16/12/2023
ef: 27/12/2023
days: 11
5.1
Creazione casi test
es: 15/12/2023
ef: 20/12/2023
days: 5
5.2
Esecuzione test
es: 20/12/2023
ef: 25/12/2023
days: 5
5.3
Risoluzione bug
es: 25/12/2023
ef: 31/12/2023
days: 6
3.3
Integrazione API
es: 05/11/2023
ef: 22/11/2023
days: 17
3.1
Sviluppo algoritmo
es: 05/11/2023
ef: 22/11/2023
days: 17
3.2
Sviluppo interfaccia
es: 05/11/2023
ef: 22/11/2023
days: 17
4.3
Test e debug codice
es: 16/12/2023
ef: 27/12/2023
days: 11
3.4
Test prototipo
es: 22/11/2023
ef: 09/12/2023
days: 17
6.1
Preparazione pacchetto
es: 01/01/2024
ef: 07/01/2024
days: 6
6.2
Documentazione finale
es: 07/01/2024
ef: 12/01/2024
days: 5
6.3
Pubblicazione app
es: 12/01/2024
ef: 17/01/2024
days: 5

5.1.3. Cammino Critico

Il **cammino critico** è: 1.1 Raccolta dei requisiti -> 1.2 Analisi e documentazione -> 1.3 Revisione dei requisiti -> 2.1 Progettazione architetturale -> 2.2 Progettazione dell'interfaccia -> 3.2 Sviluppo interfaccia -> 3.4 Test prototipo -> 4.1 Implementazione funzionalità -> 4.2 Ottimizzazione codice -> 4.3 Test e debug codice -> 5.1 Creazione casi test -> 5.2 Esecuzione test -> 5.3 Risoluzione bug -> 6.1 Preparazione pacchetto -> 6.2 Documentazione finale -> 6.3 Pubblicazione app.

Durata totale del cammino critico: 10 + 5 + 3 + 7 + 14 + 17 + 17 + 20 + 11 + 11 + 5 + 5 + 6 + 6 + 5 + 5 = **141 giorni**.

5.1.4. Matrice Compiti Responsabilità

Niccolò Pirillo: Raccolta dei requisiti, Progettazione dell'interfaccia, Sviluppo interfaccia, Implementazione funzionalità, Creazione casi test, Preparazione pacchetto.

Federico Nori: Analisi e documentazione, Progettazione architetturale, Sviluppo algoritmo, Ottimizzazione codice, Esecuzione test, Documentazione finale.

Riccardo Vendraminetto: Revisione dei requisiti, Progettazione dell'algoritmo, Integrazione API, Test e debug codice, Risoluzione bug, Pubblicazione app.

Beatrice Gigli: Progettazione integrazione API, Test prototipo, Implementazione funzionalità, Creazione casi test, Preparazione pacchetto.

Enrico Giacobbi: Sviluppo algoritmo, Sviluppo interfaccia, Ottimizzazione codice, Esecuzione test, Documentazione finale.

5.2. Dipendenze

Progettazione dipende da: Analisi dei requisiti. Non è possibile progettare l'applicazione senza prima aver raccolto e analizzato i requisiti.

Prototipazione dipende da: Progettazione. Il prototipo UI deve essere basato sulle scelte di progettazione fatte in precedenza.

Sviluppo dipende da: Prototipazione. Lo sviluppo dell'applicazione inizia dopo che il prototipo è stato verificato con lo “stakeholder”.

- La creazione del progetto Android e la configurazione delle dipendenze devono essere completate prima di poter iniziare a sviluppare l'algoritmo di estrazione dati e l'interfaccia grafica.

- Lo sviluppo dell'algoritmo di estrazione dati open-data deve essere completato prima di poter integrare le API di Google Maps, poiché queste due attività possono essere strettamente collegate.

Testing dipende da: Sviluppo. Il testing inizia dopo che lo sviluppo dell'applicazione è sostanzialmente completo.

Rilascio dipende da: Testing. Il rilascio dell'applicazione può avvenire solo dopo che il testing è completato e i bug sono stati risolti.

Nel dettaglio:

- Il task 1.2 Analisi e documentazione dipende dalla conclusione di 1.1 Raccolta dei requisiti.
- Il task 1.3 Revisione dei requisiti dipende dalla conclusione di 1.2 Analisi e documentazione.
- Il task 2.1 Progettazione architetturale può iniziare subito dopo la conclusione di 1.3 Revisione dei requisiti.
- Il task 2.2 Progettazione dell'interfaccia dipende dalla conclusione di 2.1 Progettazione architetturale.
- Il task 2.3 Progettazione dell'algoritmo dipende dalla conclusione di 2.1 Progettazione architetturale.
- Il task 2.4 Progettazione integrazione API dipende dalla conclusione di 2.1 Progettazione architetturale.
- Il task 3.1 Sviluppo algoritmo dipende dalla conclusione di 2.3 Progettazione dell'algoritmo.
- Il task 3.2 Sviluppo interfaccia dipende dalla conclusione di 2.2 Progettazione dell'interfaccia.
- Il task 3.3 Integrazione API dipende dalla conclusione di 2.4 Progettazione integrazione API.
- Il task 3.4 Test prototipo dipende dalla conclusione di 3.1 Sviluppo algoritmo, 3.2 Sviluppo interfaccia, e 3.3 Integrazione API.
- Il task 4.2 Ottimizzazione codice e 4.3 Test e debug codice dipendono dalla conclusione di 4.1 Implementazione funzionalità.
- Il task 5.2 Esecuzione test dipende dalla conclusione di 5.1 Creazione casi test.
- Il task 5.3 Risoluzione bug dipende dalla conclusione di 5.2 Esecuzione test.
- Il task 6.3 Pubblicazione app dipende dalla conclusione di 6.1 Preparazione pacchetto e 6.2 Documentazione finale.

5.3. Risorse necessarie

Per le risorse umane, abbiamo bisogno di tutti i membri del team per dedicare una certa quantità di ore a settimana per la durata del progetto. Per le risorse hardware, avremo bisogno di dispositivi performanti per lo sviluppo e il testing con Android Studio.

5.4. Allocazione del budget e delle risorse

Stipendi degli ingegneri/programmatori
Nell'ipotetico caso in cui si assumano sviluppatori a tempo pieno per 4 mesi il costo degli stipendi sarà:
5 sviluppatori * € 2.000 al mese * 4 mesi = € 40.000

Costi dell'Hardware e del software
Laptop per ogni sviluppatore: 5 sviluppatori * € 1.000 = € 5.000
Licenze software: 5 sviluppatori * € 500 = € 2.500

Hardware di test e altri costi: € 2.000
Totale: € 5.000 + € 2.500 + € 2.000 = € 9.500

Costi per viaggi e formazione
Formazione: 5 sviluppatori * € 500 = € 2.500
Costi di viaggio: € 1.000
Totale: € 2.500 + € 1.000 = € 3.500

Altri costi
Supponiamo che altri costi (affitto, riscaldamento, illuminazione, supporto tecnico, pulizie, reti e telecomunicazioni) siano pari al 50% dei costi degli stipendi.
€ 40.000 * 50% = € 20.000

Totale
€ 40.000 (stipendi) + € 9.500 (hardware e software) + € 3.500 (viaggi e formazione) + € 20.000 (altri costi) = € 73.000

Quindi, una stima approssimativa del costo totale del progetto potrebbe essere di € 73.000.

Per ammortizzare i costi è possibile procedere con lo sviluppo dell'app totalmente in smart working, eliminando gli altri costi, per un totale di:
€ 73.000 - € 20.000 = € 53.000.

5.5. Pianificazione

La programmazione delle attività [rif. 5.1] seguirà scrupolosamente le scadenze dei deliverables forniti dal docente prof. A. Cortesi [rif. 1.3].