Piano di Progetto

Smart Meal

Gruppo: Quadcore

Facci Matteo 875377
Guidobene Davide 877150
Pittini Enrico 877345
Rosin Giacomo 875724

Versione 2.0

21/01/2021

Indice

1. Introduzione

1.1 Overview del Progetto

Il progetto ha avuto inizio in data 14 Settembre 2020 e sarà terminato entro il 15 Gennaio 2021.

L'applicazione serve per migliorare l'interazione di un locale con i suoi clienti offrendo funzionalità come un messaggio di benvenuto rilevando l'ingresso del cliente, un comodo menù virtuale (particolarmente utile in periodo di pandemia) e la possibilità di chiamare un cameriere semplicemente scuotendo il dispositivo.

1.2 Deliverables del Progetto

Tutti i deliverables andranno consegnati su Moodle, entro le date designate:

- Definizione dei gruppi di lavoro (21/09/2020)
- Proposta iniziale (29/9/2020)
- Piano di Progetto (18/10/2020)
- Documento di analisi e specifica (2/11/2020)
- Piano di testing (19/11/2020)
- Documento di Progettazione (14/12/2020)
- Codice e messa in linea del sistema (15/01/2021)

1.3 Evoluzione del Progetto

Il progetto, nella sua versione di base, è incentrato su un singolo locale e sui suoi clienti. Nella sua versione più evoluta, invece, prevede la gestione di un database di tutti i locali che vogliono registrarsi.

L'applicazione potrebbe cadere in disuso in caso di un nuovo lock-down che porterebbe a una chiusura dei locali che usufruiscono del nostro servizio. In tal caso pianifichiamo lo sviluppo di un sistema di ordinazioni online con consegna a domicilio. Sebbene questo genere di funzionalità sia già coperto da diverse applicazioni, questa espansione avrebbe senso in caso l'applicazione avesse già raggiunto e fidelizzato una discreta userbase.

Se durante lo sviluppo del progetto si dovesse rivelare necessario apportare dei cambiamenti, relativamente agli obiettivi o relativamente alle attività, seguiremo il seguente iter: i membri del team scheduleranno una riunione d'urgenza per discutere delle ragioni del cambiamento. Se dalla riunione risulterà che apportare un cambiamento al progetto sia utile per renderlo migliore o, nei casi peggiori, addirittura necessario per salvarne l'esito, il team si muoverà in modo da apportare le dovute modifiche ai documenti (Piano di Progetto, Documento di analisi e specifica, …) e al codice. Nel caso in cui il cambiamento fosse drastico sarà necessario comunicarlo anche agli stakeholders interessati.

1.4 Materiali di riferimento

- slides del corso di Ingegneria del Software [CT0090] del professor Cortesi

- Appunti delle lezioni di Ingegneria del Software [CT0090] dei professori Agostino Cortesi e Alvise Spanò e del tutor Giulio Zausa
- Wikipedia (per la stesura delle definizioni)
- Documentazione ufficiale Android (https://developer.android.com/docs)
- Documentazione Nearby API (https://developers.google.com/nearby)
- Libro “Android 5 Programming by Example”, Kyle Mew
- Libro “Android Programming for Beginners”, John Horton
- Android Studio

Con l’avanzamento del progetto, il gruppo si riserva la possibilità di aggiungere ulteriori materiali di riferimento che si riterranno utili per lo svolgimento del lavoro.

1.5 Definizioni e Abbreviazioni

- Android: Sistema operativo per dispositivi mobili sviluppato da Google. In particolare, nel progetto il termine Android fa riferimento alla versione 5.0 o successive.
- Android Studio: Ambiente di sviluppo integrato che supporta i programmatori nello sviluppo e debugging del codice sorgente.
- Bug: Guasto che porta al malfunzionamento del software, tipicamente dovuto ad un errore nella scrittura del codice sorgente di un programma.
- Continuous Integration (CI): Pratica che si applica in contesti in cui lo sviluppo del software avviene attraverso un sistema di controllo versione. Consiste nell'allineamento frequente (ovvero "molte volte al giorno") dagli ambienti di lavoro degli sviluppatori verso l'ambiente condiviso.
- GitHub: Servizio di hosting per progetti software che utilizzano Git come sistema di versionamento del codice.
- Git: Software di controllo di versione distribuito, che semplifica il mantenimento di codice sviluppato da un team.
- Google Drive: Servizio di hosting web, che permette di memorizzare documenti sul cloud e di condividerli con altri utenti.
- Google Docs: Word processor web che permette ad un team di collaborare contemporaneamente alla scrittura di un documento condiviso.
- Google Meet: Applicazione di teleconferenza sviluppata da Google, che mette a disposizione oltre alla classica videochiamata anche la condivisione dello schermo dei vari partecipanti.
- JavaDoc: Formato standard dei commenti per la documentazione interna al codice Java. I commenti scritti in questo formato, oltre ad essere human-readable sono riconosciuti dagli IDE e da vari tool che permettono di generare la documentazione in modo automatico in formato HTML.
- Pair programming: Tecnica agile di sviluppo del software nella quale due programmatori lavorano insieme ad una postazione di lavoro. Uno dei due scrive il codice; l'altro svolge un ruolo di supervisione e di revisione simultanea del codice.
- Whatsapp: Applicazione di messaggistica istantanea, che permette di comunicare tramite internet.

2. Organizzazione del progetto

2.1 Modello del Processo

La nostra scelta di modello di processo è ricaduta su quello di tipo evolutivo perché abbiamo valutato fosse il più adatto al progetto scelto e ai membri del team di sviluppo. Infatti la sua struttura ci permette di monitorare agevolmente l'evoluzione dell'applicazione verso il sistema finale, caratteristica che lo rende il modello per eccellenza in termini di scalabilità ed espandibilità offrendoci un potente mezzo per prevenire errori. Errori che potrebbero essere facilmente causati dalla nostra scarsa esperienza nel campo, come potrebbe esserlo una definizione delle specifiche del progetto troppo grande o, all'opposto, troppo inferiore rispetto alle nostre possibilità.

Per quanto riguarda la riguarda la prototipazione del modello, abbiamo optato per il tipo evolutivo riservandoci di usare la prototipazione di tipo usa e getta per sperimentare le funzionalità che ci sono meno familiari o che richiedono una progettazione più attenta. Questa scelta consente al team di sviluppo di aggiungere funzionalità o apportare modifiche che non è stato possibile concepire durante la fase dei requisiti e della progettazione.

Ispirandoci all'extreme programming abbiamo deciso di adottare anche la tecnica del *pair programming* al fine di limitare i rischi di rallentamento dovuti a bug nel codice e per sostenere l'avanzamento del progetto mantenendo alta l'attenzione del gruppo durante la fase di coding.

2.2 Struttura Organizzativa

Abbiamo optato per una tipologia di team Democratico Decentralizzato per le seguenti ragioni:

- si adatta perfettamente alle piccole dimensioni del nostro team di sviluppo di 4 persone (e che quindi non potrà essere rallentato significativamente da una comunicazione orizzontale)
- la responsabilità delle decisioni sarà condivisa equamente tra tutti i componenti, portando tutti a essere maggiormente motivati a contribuire e complessivamente a un'attitudine propositiva per il progetto
- previene il rischio di conflitti tra i componenti del gruppo causati da dissensi con le decisioni di un eventuale leader

Per la struttura organizzativa abbiamo invece preferito una divisione per
progetto dal momento che la nostra azienda è da considerarsi una start-up e si occupa di progetti di natura tecnologica, che rappresentano spesso una sfida a se stante.

2.3 Interfacce Organizzative

Il team si relazionerà con:

- i professori Cortesi Agostino, Spanò Alvise e con il tutor Zausa Giulio per avere un punto di riferimento nel corso dello sviluppo;
- utenti tester per avere dei feedback sul funzionamento dell'applicazione e sulla qualità dell'esperienza utente
- i responsabili dei locali con i quali lavoreremo per lo sviluppo dell'applicazione

L’interfacciamento con i soggetti citati sarà curato da Rosin Giacomo, e avverrà prevalentemente via e-mail.

2.4 Responsabilità di Progetto

Nonostante i membri del gruppo si occuperanno in modo trasversale di tutte le attività, abbiamo deciso di nominare uno o più responsabili per ciascune di esse che, oltre a svolgere i suoi normali compiti di membro del gruppo dovrà assicurarsi che le attività sotto la sua responsabilità siano svolte regolarmente.

Technical staff
Responsabili: Guidobene Davide, Pittini Enrico
Incarico: Condurre l’analisi e lo sviluppo del codice
Natura: Programmazione, sviluppo

Software librarian
Responsabile: Rosin Giacomo
Incarico**:** Mantenere e controllare la qualità del codice, della documentazione e dei dati
Natura: Programmazione, controllo, standardizzazione

Docs designer
Responsabile: Facci Matteo
Incarico: Controllare lo stile e la completezza dei documenti
Natura: Controllo, documentazione, linguaggio, standardizzazione

Graphic designer
Responsabili: Facci Matteo, Rosin Giacomo
Incarico: Prototipare l’interfaccia utente e controllarne l’uniformità e l’usabilità
Natura: Interfaccia, standardizzazione

Testing manager
Responsabile**:** Pittini Enrico
Incarico**:** Controllare la qualità e la comprensività dei test
Natura**:** Programmazione, controllo

Minute taker
Responsabile**:** Guidobene Davide
Incarico**:** Registrare i verbali delle riunioni e tenere traccia dell’ordine del giorno
Natura**:** Controllo, linguaggio, documentazione

3. Processi Gestionali

3.1 Obiettivi e Priorità

Il nostro obiettivo è quello di realizzare l'applicazione (il cui scopo principale è descritto nell'introduzione) e la relativa documentazione restando coerenti con le specifiche già descritte in questo documento, e allo stesso tempo far si che l'applicazione abbia un'interfaccia utente e una user experience che risultino gradevoli nell'utilizzo.

Priorità:

- Rispettare le scadenze indicate;
- Consegnare un applicativo funzionante, conforme a tutti i requisiti forniti.

3.2 Assunzioni, Dipendenze, Vincoli

Le assunzioni che poniamo sono:

- Conoscenza da parte di ogni componente del gruppo del linguaggio di programmazione Java.
- Possibilità di effettuare lavoro e comunicazione da remoto per ciascun componente, quindi avere a disposizione un computer con connessione internet.
- Capacità e volontà di lavorare in team.

Le nostre dipendenze sono:

- Apprendimento programmazione in ambiente android.
- Le comunicazioni tra i vari membri del gruppo verranno effettuate interamente da remoto nel caso in cui l'emergenza coronavirus peggiorasse.

I vincoli sono:

- Linee guida professore;
- Scadenze temporali, indicate sopra (nella sezione 2).
- Gestione del tempo in modo conforme agli altri esami e impegni.

3.3 Gestione dei rischi e i loro meccanismi di monitoraggio e di controllo

Id rischio	Rischio	Categoria	Probabilità	Impatto	Soluzione adottata
1	Locale non coperto completamente dal segnale (o interferenze)	Tecnologico	50%	Media	Possiamo risolvere questa problematica facendo si che i gestori utilizzino più dispositivi all'interno del locale

2	Errori di comunicazione relative all'uso dell'app (notifica ad un cliente che non stava entrando nel locale, ma era di passaggio o andava in un locale vicino)	Tecnologico	40%	Bassa	Nel caso il dispositivo del gestore comunichi erronamente con il dispositivo di una persona non cliente del locale (ad esempio un passante) la persona in questione potrà rifiutare la richiesta inviata dal gestore.
3	Imprecisione dei sensori	Tecnologico	80%	Medio	Nell'applicazione se uno dei sensori non funziona correttamente vi è un sistema per risolvere le incomprensioni tra gestore e cliente.
4	Mantenibilità del codice scadente	Tecnologico	20%	Alta	Ci impegnamo fin dalla prima riga di codice, rendendo il codice modulare e ben strutturato.
5	Malattia di almeno un componente del gruppo	Risorse Umane	80%	Medio	Se la malattia è leggera si può continuare comunque con lo smart working. Se la malattia è più grave invece, anche se il lavoro può essere comunque svolto dagli altri componenti del gruppo, ma diminuisce la forza lavoro.
6	Problematiche/litigi	Risorse Umane	40%	Alto	Ci impegnamo ogni settimana ad effettuare meeting almeno da remoto in cui discutiamo relativamente all'evoluzione del progetto, risolvendo eventuali incomprensioni e spiegando le relative scelte

7	Requisiti non coperti dall'applicazione	Requisiti	40%	Alto	Grazie all'attenzione verso il documento dei requisiti che faremo successsivamente, ridurremmo al minimo le probabilità di questo rischio

Tabella di classificazione

In seguito è riportata la tabella di classificazione, con la probabilità sull'asse delle ascisse e l'impatto sull' asse delle ordinate, con probabilità ed impatto crescenti.

4	**6,7**			
		1	**3,5**	
	2			

3.4 Pianificazione dello staff

Nel pianificare lo staff abbiamo preso in considerazione le capacità e il background di ciascun componente del gruppo. Inoltre abbiamo sempre cercato di dividere le mansioni a coppie di persone, in modo da aumentare il controllo e la bontà del lavoro.

4. Processi tecnici

4.1 Metodi, Strumenti e Tecniche

Gli strumenti e i software da noi usati sono

- Android Studio come IDE per lo sviluppo dell'applicazione Android
- Git come sistema di versionamento del codice, utilizzando Github come servizio di hosting per memorizzare il codice dell'applicazione
- Google Drive come piattaforma per scrivere i vari documenti e condividere eventuali file non relativi al codice
- Google Meet come piattaforma per effettuare gli incontri da remoto
- WhatsApp per scambiarci informazioni e organizzarci riguardo i meeting
- Dispositivi Android per comunicare e per testare l'applicazione
- Vari laptop e computer (almeno uno per componente del gruppo) per scrivere codice e comunicare

Ci impegniamo a scrivere il codice seguendo le convenzioni del linguaggio Java ed in particolare le convenzioni per lo sviluppo di applicazioni Android.
Per standardizzare il formato del codice seguiremo la Java Style Guide di Google.

4.2 Documentazione del Software

Ci siamo preposti effettuare la documentazione in due fasi:

- quella interna al codice, in particolare usando il formato JavaDoc per descrivere ogni classe, metodo e campo, e i commenti inline per migliorare la leggibilità del codice
- quella esterna, inclusa all'interno di un documento esterno e consultabile in un qualsiasi momento per comprendere nel complesso lo sviluppo dell'applicazione.

4.3 Funzionalità di supporto al progetto

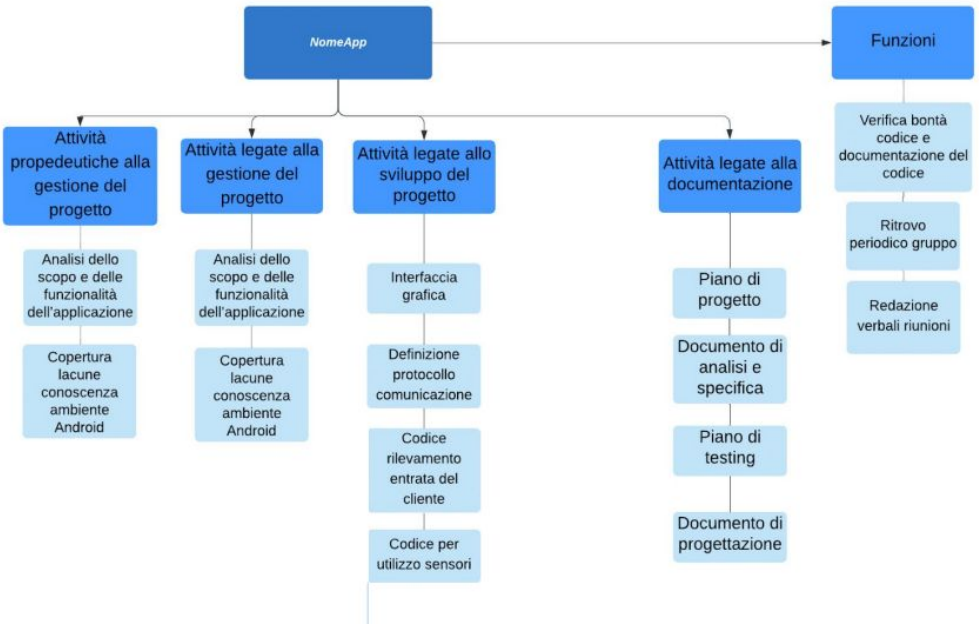
Ogni settimana scegliamo un componente del gruppo (a turno) che deve controllare la bontà del codice e della documentazione scritta.
In particolare prestiamo particolare attenzione a scrivere codice modulare, mantenibile, leggibile e scalabile, in modo che sia il più sicuro e flessibile possibile.
Inoltre sarà cura di tutti i membri del gruppo assicurarsi di associare l'annotazione `@Nullable` o `@NonNull` a ogni campo o parametro formale di ogni classe e ogni metodo Java in modo che il compilatore ci aiuti a mantenere il codice sound e privo di errori.

Utilizzeremo Git in modo da garantire che la modifica e la gestione dei file sorgenti da parte dei membri del team sia disciplinata. In particolare utilizzeremo il branch 'main' come "contenitore" per le release ufficiali del progetto, e il branch 'develop' per contenere il codice su cui lavoreremo giornalmente, e quindi potenzialmente instabile. Ogni volta che raggiungeremo una versione stabile dell'applicazione essa verrà inserita nella branch 'main' con relativo tag indicante la versione. Utilizzeremo inoltre una branch per ogni feature distinta che vogliamo sviluppare in modo da avere meno conflitti possibile e parallelizzare lo sviluppo. Queste branch devono essere quindi una ramificazione della branch 'develop', e non appena lo sviluppo della feature è terminato faremo il merge nella 'develop'.

Ci impegneremo ad effettuare commit frequenti e con una descrizione dettagliata, in modo da avere una storia dei commit ricca e facile da navigare. Inoltre faremo push e merge frequenti per mantenere gli ambienti di lavoro degli sviluppatori il più allineati possibile seguendo la pratica della Continuous Integration (CI).

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

5.1 WBS (Work breakdown structure)



Funzioni legate alla gestione del progetto

- verifica bontà codice e documentazione del codice
- ritrovo periodico gruppo
- redazione verbali riunioni

Attività propedeutiche alla gestione del progetto

- Analisi dello scopo e delle funzionalità dell'applicazione (durata:7 giorni)
- Copertura lacune conoscenza ambiente Android (durata: 21 giorni)

Attività legate alla gestione del progetto

- Analisi specifiche applicazione (durata: 19 giorni)
- Validazione applicazione (durata:15 giorni)
- Testing finale (durata: 7 giorni)

Attività legate allo sviluppo del progetto

- Interfaccia grafica (durata: 24 giorni)
- definizione protocollo comunicazione (durata: 11 giorni)
- codice che gestisce la comunicazione peer to peer (durata: 28 giorni)
- codice rilevamento entrata del cliente (durata: 18 giorni)
- codice per utilizzo sensori (durata: 24 giorni)

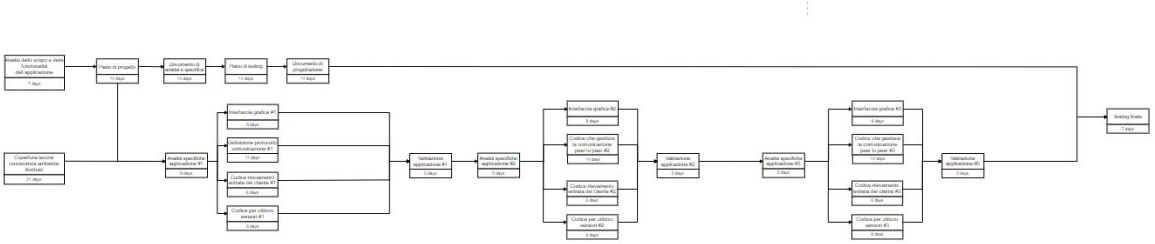
Attività legate alla documentazione

- Piano di progetto (durata: 14 giorni)
- Documento di analisi e specifica (durata: 14 giorni)

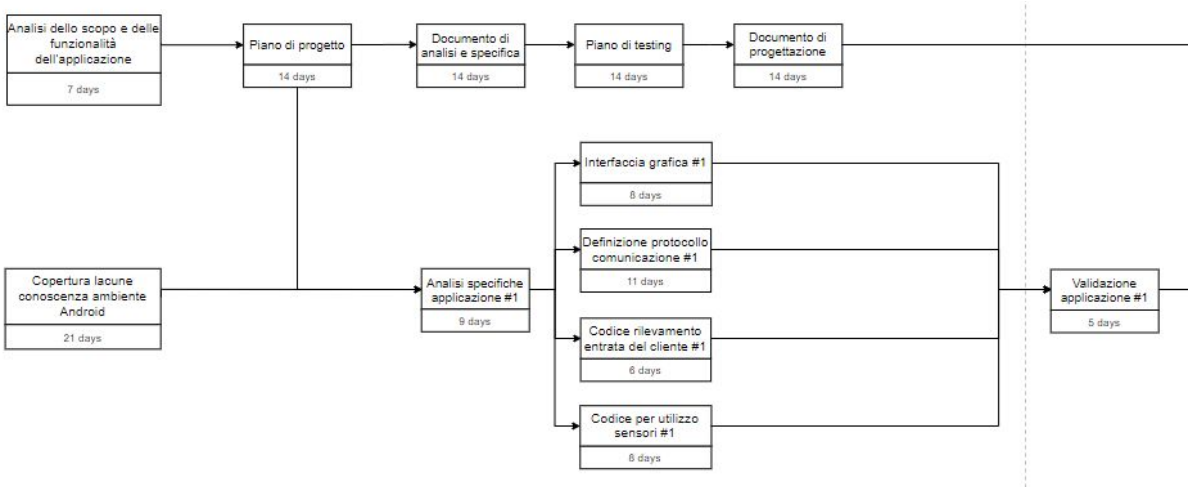
- Piano di testing (durata: 14 giorni)
- Documento di progettazione (durata: 14 giorni)

5.2 Dipendenze

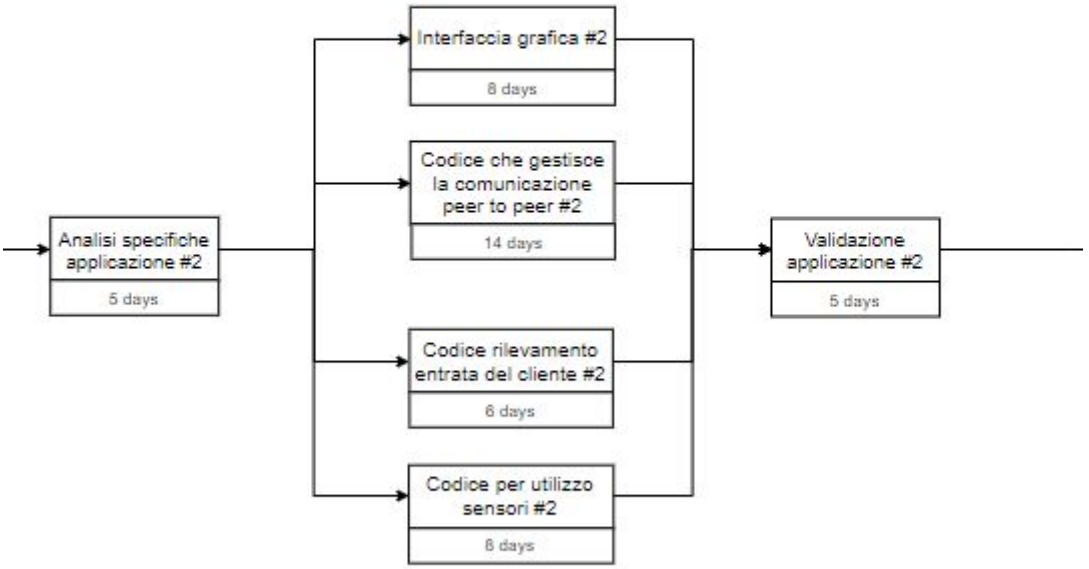
Diagramma di Pert



Parte 1



Parte 2



Parte 3

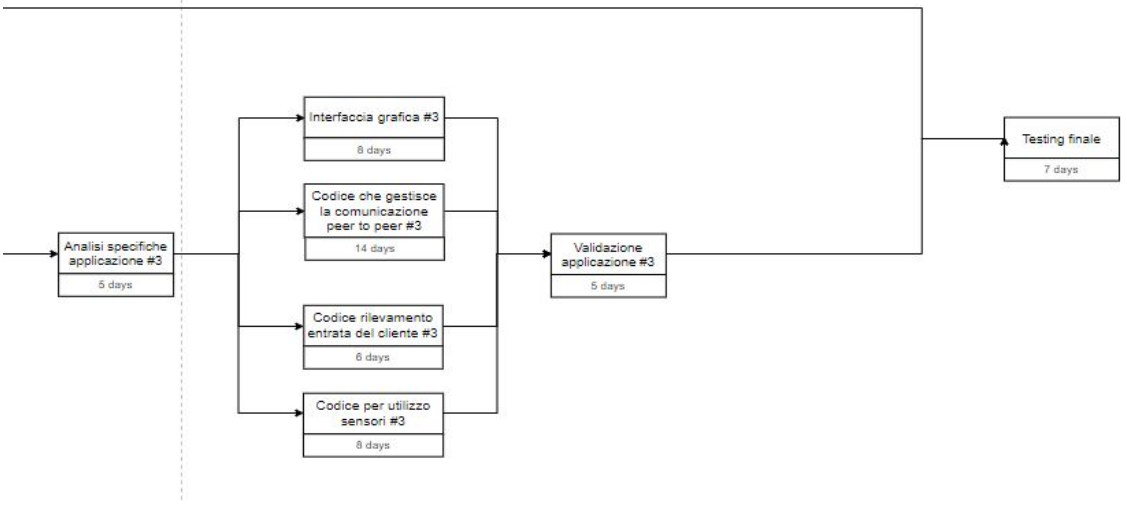
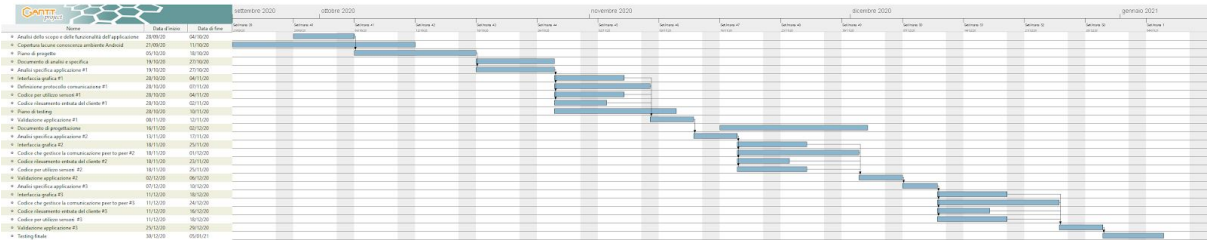


Diagramma di Gantt

Nome	Data d'inizio	Data di fine
Copertura lacune conoscenza ambiente Android	21/09/20	11/10/20
Analisi dello scopo e delle funzionalità dell'applicazione	28/09/20	04/10/20
Piano di progetto	05/10/20	18/10/20
Documento di analisi e specifica	19/10/20	27/10/20
Analisi specifica applicazione #1	19/10/20	27/10/20
Interfaccia grafica #1	28/10/20	04/11/20
Definizione protocollo comunicazione #1	28/10/20	07/11/20
Codice per utilizzo sensori #1	28/10/20	04/11/20
Codice rilevamento entrata del cliente #1	28/10/20	02/11/20
Piano di testing	28/10/20	10/11/20
Validazione applicazione #1	08/11/20	12/11/20
Analisi specifica applicazione #2	13/11/20	17/11/20
Documento di progettazione	16/11/20	02/12/20
Interfaccia grafica #2	18/11/20	25/11/20
Codice che gestisce la comunicazione peer to peer #2	18/11/20	01/12/20
Codice rilevamento entrata del cliente #2	18/11/20	23/11/20
Codice per utilizzo sensori #2	18/11/20	25/11/20
Validazione applicazione #2	02/12/20	06/12/20
Analisi specifica applicazione #3	07/12/20	10/12/20
Interfaccia grafica #3	11/12/20	18/12/20
Codice che gestisce la comunicazione peer to peer #3	11/12/20	24/12/20
Codice rilevamento entrata del cliente #3	11/12/20	16/12/20
Codice per utilizzo sensori #3	11/12/20	18/12/20
Validazione applicazione #3	25/12/20	29/12/20
Testing finale	30/12/20	05/01/21



5.3 Risorse Necessarie

Le risorse necessarie per la realizzazione del progetto sono:
Risorse umane: i 4 membri del team, che si occupano in toto della gestione dell'applicazione, dello sviluppo dell'applicazione e della stesura della documentazione.
Il lavoro dei membri del team durerà un totale di 3 mesi/uomo.

Risorse hardware: almeno un computer per ogni componente del gruppo, dispositivi Android per testare l'applicazione
Risorse software: il gruppo utilizza Android Studio, Google Meet, Google Docs, WhatsApp, Git e GitHub.

5.4 Allocazione del Budget e delle Risorse

Il budget allocato al nostro progetto è relativo solamente al tempo dedicato dai membri del team, dato che non è necessario l'acquisto di ulteriore strumentazione oltre a quelle che già avevamo e i software utilizzati sono tutti disponibili gratuitamente.

Le durate delle varie attività sono definite nel paragrafo 5.1 e considerando la parallelizzazione di alcune di esse stimiamo che il progetto richiederà un tempo totale T di 107 giorni.
Considerando che il valore lavoro per persona LP è 12€/h e che ogni giorno il tempo di lavoro medio di ciascuno degli N=4 membri del gruppo è OpG=1.5h/giorno otteniamo un budget totale di

$$B_{tot} = T*N*OpG*LP = 7700€$$

5.5 Pianificazione

Le deadlines del progetto sono già state citate precedentemente nel punto **1.2**.

Le milestones relative a questo progetto coincideranno con il completamento dei vari documenti (Piano di Progetto, Documento di analisi e specifica, Piano di testing, Documento di Progettazione), con il termine di ogni ciclo del modello evolutivo che adottiamo e al completamento della versione finale dell'applicazione.