**F2F – Families-To-Families Sharing Care of Children  
MIUR FISR 2020**

**Progetto di Ingegneria del Software 2021/22  
Università Ca’ Foscari Venezia**

**Piano di progetto**

versione 1.0  
 **Black Buffalos**

15 ottobre 2021

*Informazioni sul Documento*

| MUR FISR 2020 |  | Acronimo | | | F2F | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Coordinatore Progetto | Prof. Agostino Cortesi | | | | | | | |
| Deliverable | Piano di Progetto | | | | | | | |
| Data di Consegna | 15/10/2021 | |  | | |  | |  |
| Team Leader | Giovanni Costa | | |  | | | 880892@stud.unive.it | |
| Team members | Massimo Cailotto 880873@stud.unive.it  Filippo Di Gennaro 882795@stud.unive.it Matteo Minardi 880895@stud.unive.it  Andrea Munarin 879607@stud.unive.it | | | | | | | |

*Document History*

| Version | Issue Date | Stage | Changes | Contributors |
| --- | --- | --- | --- | --- |
| 1.0 | 15/10/2021 | Draft | First version | All members of the team |
| 1.1 |  | Draft |  |  |
| 2.0 |  | Final |  |  |

**Indice**

[***1.***](#bookmark=id.677qhrk17z6r) ***Introduzione*** **4**

[1.1 Overview del progetto](#bookmark=id.mm2caagoh44v) 4

[1.2 Deliverables del progetto](#bookmark=id.xkw41qra4203) 4

[1.3 Evoluzione del progetto](#bookmark=id.mwlduby4iy87) 5

[1.4 Materiali di riferimento](#bookmark=id.t0gb5u8rfghs) 5

[1.5 Definizione e abbreviazioni](#bookmark=id.ku2abboi6qez)5

[***2.***](#bookmark=id.pwugrw2lgqye) ***Organizzazione del progetto*** 8

[2.1](#bookmark=id.rys884kgtnn) Modello del processo 8  
[2.2 Struttura organizzativa](#bookmark=id.yh0n6571y666) 9  
[2.3 Interfacce Organizzative](#bookmark=id.d41ot0atrcx3) 10  
[2.4 Responsabilità di Progetto](#bookmark=id.lyaqkwjez0d3)11

[***3.***](#bookmark=id.iekd4v2nl9po) ***Descrizione dei Processi gestionali*** 12

[3.1](#bookmark=id.udmfsg4pn21k) Obiettivi e Priorità 12

[3.2 Assunzioni, Dipendenze, Vincoli](#bookmark=id.tvru09b85ldo) 14

[3.3 Gestione dei rischi](#bookmark=id.9waeztrpefmb) 14

[3.4 Meccanismi di monitoraggio e di controllo](#bookmark=id.s2s8zzqhvwzy) 18

[3.5 Pianificazione dello staff](#bookmark=id.8m6ut69sxoc3)19

[***4.***](#bookmark=id.6i4udqp8jw6n) ***Descrizione dei Processi tecnici*** 19

[4.1 Metodi, Strumenti e Tecniche](#bookmark=id.3eb9gm2lhmif) 19

[4.2 Documentazione del software](#bookmark=id.sndkzxbme92i) 20

[4.3 Funzionalità e supporto al progetto](#bookmark=id.jy9filcqv4b9) 21

[**5.**  **Pianificazione del lavoro, delle risorse umane e del budget**](#bookmark=id.22y6vegsrxsq) **22**

[5.1](#bookmark=id.6uif4k3u1pd) WBS (Work breakdown structure) 22

[5.2 Dipendenze](#bookmark=id.fuphmrduotz) 28

[5.3 Risorse necessarie](#bookmark=id.hfq37927uc2f) 31

[5.4 Allocazione del Budget e delle Risorse](#bookmark=id.p6is709ssesm) 32

[5.5 Pianificazione](#bookmark=id.53uez9sre4jg) 33

[**6. Riferimenti**](#bookmark=id.heww2x1f4yzp) **34**

1. **Introduzione**

* 1. **Overview del progetto**

Viene richiesto lo svolgimento delle fasi di analisi, progettazione, sviluppo e testing al fine di migliorare ed ampliare con un app Android una soluzione innovativa già esistente basata sul mutuo-aiuto e sulla solidarietà tra famiglie, Families Share.

*Families Share è un’iniziativa dal basso in cui le famiglie condividono tempo e attività legate alla cura dei bambini, alla genitorialità, al dopo-scuola e al tempo libero, con il supporto di un’app open source che facilita la pianificazione e la gestione dei turni. Families Share promuove nuove idee e modalità per condividere i bisogni di cura all’infanzia e trovare soluzioni flessibili per la conciliazione vita privata – lavoro.*

*Laboratori di dopo-scuola, intere settimane di vacanze estive o invernali, pedibus sono tutte possibili attività in cui i bambini possono essere coinvolti in modo creativo e divertente, partecipando con i loro genitori, e dove le mamme e i papà possono contribuire con le loro competenze personali e passioni!*

*Questo approccio è stato sviluppato e si è dimostrato efficace grazie al progetto europeo Families Share finanziato dal programma H2020 e sperimentato in 6 città in 4 diversi Stati. Il progetto ha visto coinvolti genitori e figli di età compresa tra 3 e 11 anni che vivono negli stessi quartieri nei 6 cosiddetti CityLabs (Bologna, Györ, Kortrijk, Salonicco, Trento, Venezia).*

L’applicazione che si andrà a sviluppare si fonda sull’idea di trovare delle soluzioni per risolvere problemi generati dalla pandemia di Covid-19, focalizzandosi sul fornire assistenza in maniera più semplice e diretta per le famiglie in difficoltà.

Si vuole quindi fornire una piattaforma di condivisione di oggetti, documenti multimediali oppure tempo per fornire sostegno ai genitori, l'intrattenimento ai bambini e aiuto per l’instaurazione di amicizie o legami profondi per entrambi

Si parlerà più approfonditamente dell’applicazione nella sezione 3.1.

* 1. **Deliverables del progetto**

Le scadenze del progetto sono state programmate nel seguente modo:

* Proposta iniziale: 29/9/2021;
* Piano di Progetto: 15/10/2021;
* Documento dei Requisiti: 25/10/2021;
* Piano di testing: 14/11/2021;
* Documento di Progettazione: 30/11/2021;
* Versione 1.0 del codice sorgente: 15/12/2021;
* Versione 1.1. del codice e allineamento documentazione: 15.01.2022;

* 1. **Evoluzione del progetto**

Il team si impegna a produrre codice modulare e scalabile, così da avvantaggiare ogni possibile evoluzione del progetto, compresa la manutenzione, l’aggiunta di funzionalità e la rimozione di quelle presenti.

Inoltre, sempre in vista di modifiche future, i vari sorgenti e il progetto stesso saranno opportunamente documentati.

Qui vengono suggerite alcune possibili evoluzioni:

* Aggiunta di una modalità intrattenimento, con giochi interattivi collettivi per i membri del gruppo
* Implementazione di funzioni ad hoc per il supporto a persone anziane
* Realizzazione di un sistema di attività e condivisione basato sulla geolocalizzazione e trusted user

* 1. **Materiali di riferimento**

Durante l’interezza del progetto il team farà appoggio sul materiale presente sulla piattaforma Moodle del corso.

Per quanto riguarda la parte di sviluppo software invece verranno utilizzati ampiamente le documentazioni e le guide di [Android](https://developer.android.com/docs), per quanto riguarda la parte front end, e di [MongoDB](https://docs.mongodb.com/guides/) e [Node.Js](https://nodejs.org/en/docs/guides/), per quanto riguarda la parte di backend con la gestione del database (già in parte sviluppata dal team di Families Share)

Il link per accedere al repository pubblico di Github è:

<https://github.com/MrCosta57/progetto_ing_del_software_2021>

* 1. **Definizioni e Abbreviazioni**

Qui sono elencati i vari termini tecnici e parole chiave che si utilizzano all’interno di questa documentazione di progetto, correlati da una breve descrizione:

* **Front-end** : sezione dell’applicazione con cui l’utente interagisce direttamente. Si occupa della gestione dei flussi di dati in input da parte degli utilizzatori e della visualizzazione dei risultati delle operazioni effettuate.
* **Back-end** : questa parte salva ed elabora gli input di dati che riceve e fondamentalmente svolge tutte quelle operazioni che sono necessarie per un corretto funzionamento dell’applicazione, al termine delle quali invia i risultati al front-end in modo che possa mostrare ciò che è stato richiesto
* **Android** : Android è un sistema operativo mobile basato su una versione modificata del [kernel Linux](https://it.wikipedia.org/wiki/Linux_(kernel)) e altri software open source, progettato principalmente per dispositivi mobili touchscreen come smartphone e tablet.

È un software gratuito e open source ed il suo codice sorgente è noto come Android Open Source Project (AOSP)

Fonte: [Wikipedia](https://it.wikipedia.org/wiki/Android)

* **JavaScript** : JavaScript è un [linguaggio di programmazione](https://it.wikipedia.org/wiki/Linguaggio_di_programmazione) [orientato agli oggetti](https://it.wikipedia.org/wiki/Programmazione_orientata_agli_oggetti) e agli [eventi](https://it.wikipedia.org/wiki/Programmazione_a_eventi), comunemente utilizzato nella [programmazione Web](https://it.wikipedia.org/wiki/Programmazione_Web) [lato client](https://it.wikipedia.org/wiki/Lato_client) (esteso poi anche al [lato server](https://it.wikipedia.org/wiki/Lato_server)) per la creazione, in [siti web](https://it.wikipedia.org/wiki/Sito_web) e [applicazioni web](https://it.wikipedia.org/wiki/Applicazione_web), di effetti dinamici [interattivi](https://it.wikipedia.org/wiki/Interattivit%C3%A0) tramite [funzioni](https://it.wikipedia.org/wiki/Funzione_(informatica)) di [script](https://it.wikipedia.org/wiki/Script) invocate da eventi innescati a loro volta in vari modi dall'utente sulla [pagina web](https://it.wikipedia.org/wiki/Pagina_web) in uso ([mouse](https://it.wikipedia.org/wiki/Mouse), [tastiera](https://it.wikipedia.org/wiki/Tastiera_(informatica)), caricamento della pagina ecc...).  
  Fonte: [Wikipedia](https://it.wikipedia.org/wiki/JavaScript)
* **Node.Js** : Node.js è un [runtime system](https://it.wikipedia.org/wiki/Runtime_system) [open source](https://it.wikipedia.org/wiki/Open_source) multipiattaforma [orientato agli eventi](https://it.wikipedia.org/wiki/Programmazione_a_eventi) per l'esecuzione di codice [JavaScript](https://it.wikipedia.org/wiki/JavaScript), costruita sul [motore JavaScript V8](https://it.wikipedia.org/wiki/V8_(motore_JavaScript)) di [Google Chrome](https://it.wikipedia.org/wiki/Google_Chrome).  
  In origine JavaScript veniva utilizzato principalmente [lato client](https://it.wikipedia.org/wiki/Lato_client). In questo scenario gli script JavaScript, generalmente incorporati all'interno dell'HTML di una pagina web, vengono interpretati da un motore di esecuzione incorporato direttamente all'interno di un [Browser](https://it.wikipedia.org/wiki/Browser). Node.js consente invece di utilizzare JavaScript anche per scrivere codice da eseguire [lato server](https://it.wikipedia.org/wiki/Lato_server), ad esempio per la produzione del contenuto delle pagine web dinamiche prima che la pagina venga inviata al [Browser](https://it.wikipedia.org/wiki/Browser) dell'utente. Node.js in questo modo permette di implementare il cosiddetto paradigma "JavaScript everywhere" (JavaScript ovunque), unificando lo sviluppo di applicazioni Web intorno ad un unico linguaggio di programmazione (JavaScript).  
  Fonte: [Wikipedia](https://it.wikipedia.org/wiki/Node.js)
* **Client-Server** : In [informatica](https://it.wikipedia.org/wiki/Informatica) il termine sistema client-server (letteralmente cliente-serviente) indica un'[architettura di rete](https://it.wikipedia.org/wiki/Architettura_di_rete) nella quale genericamente un [computer](https://it.wikipedia.org/wiki/Computer) [client](https://it.wikipedia.org/wiki/Client) o [terminale](https://it.wikipedia.org/wiki/Terminale_(informatica)) si [connette](https://it.wikipedia.org/wiki/Connessione_(informatica)) ad un [server](https://it.wikipedia.org/wiki/Server) per la fruizione di un certo servizio, quale ad esempio la condivisione di una certa [risorsa](https://it.wikipedia.org/wiki/Risorsa_informatica) [hardware](https://it.wikipedia.org/wiki/Hardware)/[software](https://it.wikipedia.org/wiki/Software) con altri client, appoggiandosi alla sottostante architettura [protocollare](https://it.wikipedia.org/wiki/Protocollo_di_rete).  
  Più semplicemente, i sistemi client/server sono un'evoluzione dei sistemi basati sulla condivisione semplice delle [risorse](https://it.wikipedia.org/wiki/Risorsa_informatica): la presenza di un [server](https://it.wikipedia.org/wiki/Server) permette ad un certo numero di client di condividerne le risorse, lasciando che sia il server a gestire gli accessi alle risorse per evitare conflitti di utilizzazione tipici dei primi [sistemi informatici](https://it.wikipedia.org/wiki/Sistema_informatico)  
  Fonte: [Wikipedia](https://it.wikipedia.org/wiki/Sistema_client/server)
* **Android Studio** : Android Studio è un ambiente di sviluppo integrato (IDE) per lo sviluppo per la piattaforma Android.  
  Fonte: [Wikipedia](https://it.wikipedia.org/wiki/Android_Studio)
* **MongoDB** : MongoDB è un [DBMS](https://it.wikipedia.org/wiki/Database_management_system) non relazionale, [orientato ai documenti](https://it.wikipedia.org/wiki/Base_di_dati_orientata_al_documento). Classificato come un database di tipo [NoSQL](https://it.wikipedia.org/wiki/NoSQL), MongoDB si allontana dalla struttura tradizionale basata su tabelle dei [database relazionali](https://it.wikipedia.org/wiki/Database_relazionali) in favore di documenti in stile [JSON](https://it.wikipedia.org/wiki/JSON) con schema dinamico (MongoDB chiama il formato [BSON](https://it.wikipedia.org/wiki/BSON)), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce.  
  Fonte: [Wikipedia](https://it.wikipedia.org/wiki/MongoDB)
* **MQL** : è il linguaggio di query (interrogazione) del database MongoDB. É molto ricco ed inoltre supporta tutte le operazioni CRUD (Create, Read, Update, Delete) sui dati
* **GitHub** : GitHub è un servizio di hosting per progetti software,, ma offre molte più funzionalità. GitHub fornisce un’interfaccia grafica basata sul web, il controllo dell'accesso e diverse funzionalità di collaborazione, come strumenti di base per la gestione delle attività per ogni progetto
* **Repository** : può essere considerato un archivio o un deposito di informazioni, accessibili più o meno liberamente e spesso raggiungibili tramite Internet
* **ES** (earliest start time) : indica il giorno minimo di inizio dell’attività a partire dal minimo tempo necessario per le attività che precedono
* **EF** (earliest finish time) : dato ES e la durata dell’attività indica il minimo giorno in cui l’attività può terminare
* **LF** (latest finish time) : il giorno massimo in cui l’attività deve finire senza creare ritardo alle attività che dipendono da lui
* **LS** (latest start time) : dato LF e la durata dell’attività indica il giorno massimo in cui l’attività deve iniziare senza provocare ritardi alle attività che dipendono da lui.

1. **Organizzazione del progetto**

* 1. **Il modello del processo**

La metodologia adottata per portare a termine il progetto è quella di tipo Agile.

Alcuni punti fondamentali di questa metodologia di sviluppo sono:

* le persone e le interazioni sono più importanti dei processi e degli strumenti (relazioni e la comunicazione tra gli attori di un progetto software sono la sua miglior risorsa)
* è più importante avere software funzionante che documentazione (vengono rilasciate nuove versioni del software ad intervalli frequenti, e viene mantenuto il codice semplice e avanzato tecnicamente, riducendo la documentazione al minimo)
* bisogna collaborare con i clienti al di là del contratto (la collaborazione diretta offre risultati migliori dei rapporti contrattuali)
* bisogna essere pronti a rispondere ai cambiamenti più che aderire al progetto (il team vuole essere più flessibile possibile per adattarsi ai cambiamenti in corso d’opera, senza rimanere ancorato ad un progetto che potrebbe bloccarsi e non portare più ai risultati desiderati)

Questa strategia è sembrata la più consona, descrivendo al meglio l’idea dei vari membri del team, e quella più appropriata per la buona riuscita del progetto, essendoci una stretta collaborazione tra gruppo e attori.

Inoltre, si vuole dare una libertà controllata ai membri del team, lasciandoli lavorare in autonomia, ma a delle task prestabilite dall’intero team

Ogni settimana vengono pianificati i task da portare a termine, da finire, rinviati e quant’altro. Durante questi incontri di pianificazione e aggiornamento, inoltre, le attività vengono schedulate considerando un margine per gli imprevisti e un ordine da rispettare nella sequenza di operazioni da svolgere con eventuali sovrapposizioni, anticipazioni o posticipazioni.

Come supporto all’organizzazione dei task vengono utilizzati tool come Jira, che consente il tracciamento degli errori e la gestione dei progetti sviluppati proprio con una metodologia agile come quella utilizzata, e Trello per la manipolazione e gestione dei task assegnati al team.

Ogni attività da portare a termine sarà assegnata ad un membro del team e/o al responsabile di quel settore

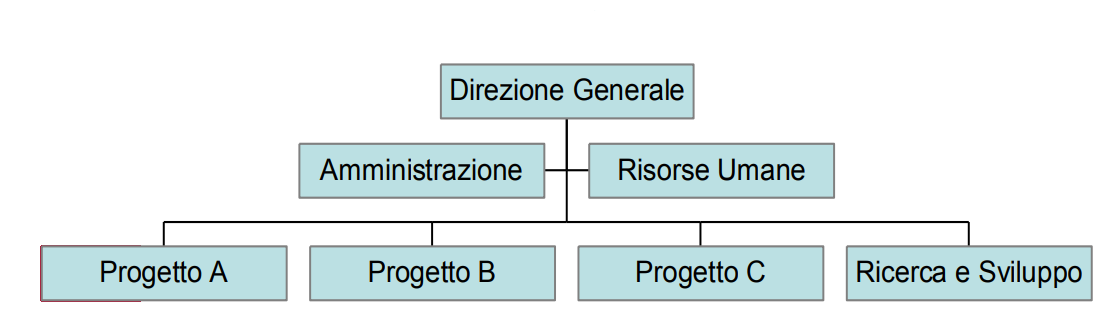
Dividiamo il progetto nelle seguenti fasi centrali:

1. Analisi e organizzazione del progetto
2. Progettazione del software
   1. algoritmo per diffusione contagi
   2. gestione green pass
   3. mercatino digitale
3. Creazione dell’app
4. Test delle singole componenti
5. Test del sistema intero
6. Migliorie all’interfaccia grafica ed eventuali risoluzioni dei problemi
7. Conclusione del progetto e raggiunta della versione finale dell’applicazione

* 1. **Struttura organizzativa**

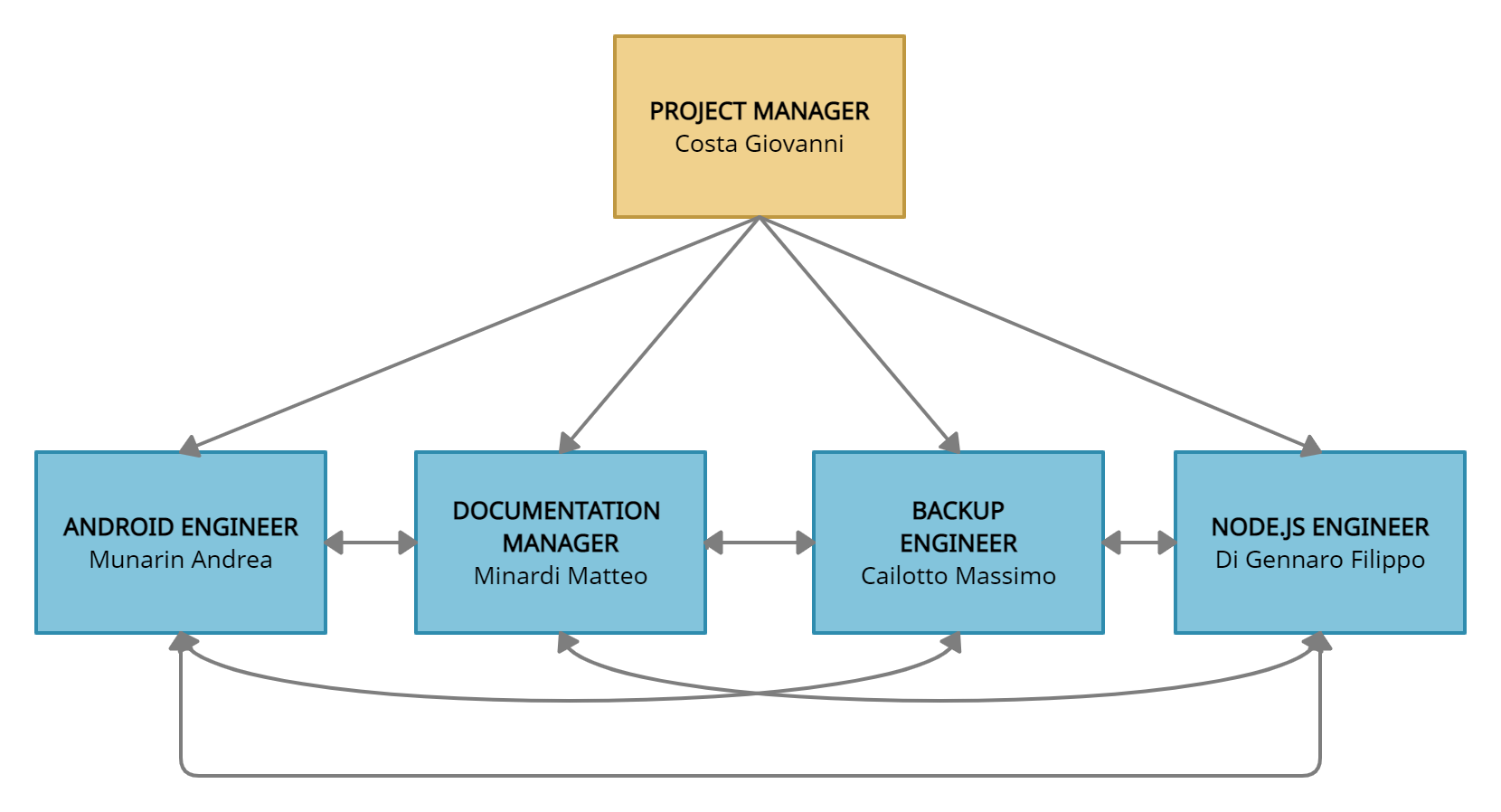
La tipologia di organizzazione che si è scelto di adottare per questo progetto è la Struttura Organizzativa per Progetto. Le principali motivazioni per cui si è deciso di utilizzare questa struttura sono le seguenti:

* Presenza di un project manager che faciliti l’interazione all’interno del gruppo e la coordinazione autoritaria di attività e risorse
* Ottimale per la situazione di un esame universitario (singolo progetto del team)



*Esempio di Organizzazione per progetto*

La tipologia di team scelta è quella del team **controllato decentralizzato**, mantenendo così una figura di riferimento e amministrativa capace di interfacciarsi con più elementi della squadra e non, e che grazie alla sua visione possa dare ad ognuno precise istruzioni eliminando eventuali ambiguità. Il leader indica le attività da portare a termine e i task da eseguire, il team decide come approcciare e risolvere i task assegnati. La comunicazione è sempre **orizzontale** tra i membri della squadra rendendo tutti gli elementi coordinati e sempre aggiornati con il progresso del progetto, e **verticale** con il leader creando così una gerarchia strutturale. Ogni persona di ogni livello avrà poi un ruolo specifico a cui attenersi con dei compiti specifici da svolgere.

*Esempio di team controllato decentralizzato*

* 1. **Interfacce organizzative**

Durante la realizzazione del progetto, per il team sicuramente sarà necessario doversi interfacciarsi con altre entità esterne, in particolare:

* Per il **monitoraggio** ed i feedback sul progetto, il team si interfaccerà con il Professor A. Cortesi periodicamente. In particolare, il membro del team che comunicherà con questa entità sarà il project manager.
* Per la **consultazione** durante le fasi di sviluppo del progetto, il team si interfaccerà con il Professor A. Spanò e con la tutor del corso M. Di Campi. Nello specifico, i membri del team che saranno più in relazione con questa entità saranno il software librarian e il technical staff.

* 1. **Responsabilità di Progetto**

In questa sezione vengono elencati i vari ruoli assegnati ad ogni membro del team, con una descrizione delle proprie responsabilità

* Project Manager
  + Componente: Giovanni Costa
  + Ruolo: pianifica, coordina e supervisiona le attività del team per il progetto corrente durante le varie settimane, accertandosi che ogni membro del gruppo svolga i compiti a lui assegnati; in più, si assicura che i deliverables vengano consegnati entro le date previste e siano completi. Inoltre si occupa di gestire le dinamiche del team qualora si incorra in uno o più dei rischi citati (sezione 3.3)
* Android engineer
  + Componente: Andrea Munarin
  + Ruolo: Si occupa della gestione e della revisione del codice prodotto dal team riguardante la parte di Android. Deve assicurarne la correttezza, l’ottimizzazione massima, la leggibilità, verificando infine, che la documentazione relativa, cioè i commenti, siano coerenti e vengano aggiunti ed aggiornati
* NodeJS engineer
  + Componente: Filippo Di Gennaro
  + Ruolo: Si occupa di tutto ciò che riguarda il codice back-end dell’applicazione. In questo è compreso anche la gestione del database NoSQL e la verifica della correttezza e efficienza delle query effettuate
* Documentation Manager
  + Componente: Matteo Minardi
  + Ruolo: Il suo obiettivo è quello di mantenere aggiornata tutta la documentazione relativa al progetto. Deve gestire i vari documenti in tutte le loro versioni e controllarne la coerenza con il lavoro svolto. Insieme ai vari engineer gestisce anche la documentazione del codice, controllando che tutti i commenti siano corretti.
* Backup Engineer
  + Componente: Massimo Cailotto
  + Ruolo: supporta il project manager nella gestione del team e lo sostituisce qualora sia impossibilitato ad adempiere al suo ruolo. È anche responsabile della validazione del software, ovvero sarà colui che darà il via libera per l’approvazione (interna la team) del codice scritto

1. **Descrizione dei Processi Gestionali**

* 1. **Obiettivi e priorità**

L’obiettivo primario è quello di creare delle migliorie da apportare all’applicazione, ampliandone le funzionalità già presenti. Parte dell’obiettivo consiste nel mantenere la struttura originale senza stravolgere o perdere di vista lo scopo per cui è stata creata.  
Il risultato finale dovrà garantire un alto livello sia dal punto di vista funzionale che qualitativo, rendendo dunque il tutto efficiente ed efficace.

Funzioni principali:  
Durante lo svolgimento del progetto, il team lavorerà per il raggiungimento dei seguenti obiettivi:

1. **Gestione e tracciamento per emergenza Covid19**

Inserimento del green pass nel proprio profilo, e in quello del bambini, con la possibilità di renderlo obbligatorio per alcune attività

Funzionalità correlate:

* 1. Tracing dei contagi

Ogni utente utente può segnalare nell’app se un membro della famiglia è stato contagiato da covid19; in tal caso tutti gli utenti che hanno partecipato ad attività insieme al malato negli ultimi 3/4 giorni vengono informati con una notifica prioritaria

1. **Bacheca di condivisione**

Creazione di una bacheca per la condivisione di vario materiale in modo pubblico tra i membri di uno stesso gruppo (poesie, appunti, testi, foto e video)

1. **Mercatino digitale**

Ogni utente può condividere un oggetto (giocattolo, libro, alimento, ecc) o decidere di cambiarlo con un'altro che richiede, anche come prestito.

Gli utilizzatori possono cercare i beni per categorie o mediante search bar

Funzioni Secondarie:  
Altri traguardi secondari che la squadra si è posta e che ritiene siano di rilevante importanza sono:

1. **Chat dedicata**

Creazione di una chat dedicata per ogni attività, accessibile e visibile solo da chi decide di partecipare e dai responsabili

1. **Migliore integrazione dei luoghi**

Connessione mediante API di Google Maps per la ricerca dei punti di interesse, con annessa apertura di un’app di navigazione al momento della selezione del luogo

1. **Orari di disponibilità**

Ogni utente può indicare i propri orari di disponibilità generali in modo da informare tutte le persone presenti nei suoi stessi gruppi

1. **Dark Mode**

Inserimento della modalità scura dell’app

Bug Fixes:  
Infine ci sono una serie di modifiche, di minore impatto, da apportare a sezioni già implementate, ma che si ritiene saranno utili per garantire il desiderato livello di qualità. È stato deciso che ci si occuperà delle seguenti questioni:

* Modificare la ripetizione delle attività in modo che sia più personalizzabile
* Controllo che i bambini inseriti abbiano una data di nascita precedente alla data corrente
* Aggiungere l’auto-eliminazione di un evento e della sua chat dopo un numero prefissato di giorni trascorsi dal suo termine

Si è scelto di dare maggiore rilevanza ed obbligatorietà alle tre funzioni descritte nel dettaglio nella prima sezione. Proprio queste sono state considerate dal team di fondamentale importanza perché specifiche e congrue al periodo storico attuale, riguardo condivisione digitale e di beni fisici per alcuni non economicamente accessibili, con controlli per il contenimento dei contagi.

Si è presunto che i punti più problematici saranno quelli contenuti nella sezione iniziale e la porzione inerente alla chat dedicata.

È stato deciso di non approfondire gli aspetti legati a servizi per anziani, per famiglie con figli grandi, e servizi ricorrenti poiché ritenuti in larga parte gestibili con la creazione di eventi “standard” e organizzazione mediante programmi di messaggistica o chat interna all’app.

Siccome l’applicazione arriverà a gestire grandi quantità di dati personali, molti di essi anche sensibili, uno dei focus a cui verrà dedicata particolare attenzione sarà quello di rendere il tutto scalabile, mantenibile in aggiornamenti futuri, e sicuro da eventuali attacchi o anomalie.

* 1. **Assunzioni, Dipendenze, Vincoli**

**Assumiamo** che tutti i membri del team siano sempre disponibili nei giorni scelti per svolgere il proprio lavoro, sia singolarmente che negli incontri di gruppo, dove vengono discusse le decisioni relative ad alcune componenti del progetto e sono quindi necessarie le opinioni di tutti i responsabili dei propri ambiti.

Per quanto riguarda le dipendenze è **necessario** che i componenti del progetto conoscano i vari linguaggi utilizzati per lo sviluppo dell’applicativo: Java (appreso durante il secondo anno universitario), Android (insegnato durante le lezioni frontali del corso), NodeJs (Discusso a lezione + supporto del Tutor), e conoscenza di ambienti NoSQL, in particolare MongoDB e la sua libreria JavaScript Mongoose (usata per la modellazione di oggetti da database a NodeJS e viceversa).

Parlando di **vincoli,** troviamo fondamentale specificare che non è possibile consegnare il materiale, ovvero i vari documenti e fasi dell’applicazione, oltre alle date di scadenze prefissate. Tutto questo per evitare che il ritardo si propaghi anche nelle consegne successive, non garantendo affidabilità e coerenza con il presente documento.

* 1. **Gestione dei rischi**

La gestione dei rischi è una componente fondamentale nello sviluppo del progetto, per questo è stato deciso di adottare un approccio di “**Strategia preventiva**” nella quale vengono precedentemente analizzati tutti i possibili rischi, se ne valutano probabilità di verificarsi e conseguenze, e si stabilisce un ordine di importanza. In questo modo è possibile gestirli in modo ordinato ed efficace

Rischi relativi ai requisiti (R):

* **Incapacità di portare a termine funzionalità principali:** Rischio di non riuscire a sviluppare pienamente le feature principali descritte negli obiettivi primari.   
  Le possibili cause di questo rischio possono derivare dalla difficoltà nella realizzazione di una delle funzionalità, rischiando di non avere tempo per realizzare le altre, o dall’impossibilità di realizzare una sua certa componente a causa di mancanza di elementi necessari per il suo sviluppo.

Azione #5: decidere in tempo se è conveniente rimuovere la funzione dall’elenco degli obiettivi.

* **Incapacità di portare a termine funzionalità secondarie:** Rischio di non riuscire a sviluppare pienamente o in parte le funzionalità secondarie descritte negli obiettivi secondari.   
  Questo può avvenire se viene dedicato troppo tempo nella realizzazione delle funzionalità principali.   
  Sono definite “funzionalità secondarie” proprio perché, data la loro minore priorità, è già stato previsto che alcune potrebbero non venire realizzate in tempo.

Azione #3: Rimuovere una funzionalità secondaria dagli obiettivi.

* **Incapacità di portare a termine bug fixes:** Rischio di non riuscire a realizzare i vari bug fixes dell’applicativo.   
  I bug fixes degli elementi già presenti nell'applicazione originale sono le attività meno importanti tra tutti gli obiettivi del team, quindi è già stata presa in considerazione l’idea di rimuoverne alcuni nel caso in cui le funzionalità elencate in precedenza necessitino più lavoro.

Azione #1: Rimuovere la correzione di alcuni bug

* **Snaturazione dell’idea originale di progetto dell’applicazione:** Rischio, durante lo sviluppo delle funzionalità aggiuntive, di perdere di vista il vero scopo dell’applicativo originale, non mantenendo la struttura prevista dagli sviluppatori di Family Share

Azione #10: Svolgere una riunione del Team per decidere come procedere e prima di effettuare eventuali modifiche alle funzionalità

Rischi relativi alle risorse umane (RU):

* **Mancanza di qualche membro del team**: Rischio di dover rimuovere un componente del team dal progetto per svariati motivi quali per esempio abbandono del corso, problemi personali

Azione #9: rivalutare i ruoli dei membri rimasti assegnando a ciascun membro una parte dei compiti del componente mancante.

* **Malattia per qualche membro del team**: Assenza temporanea (tempo variabile 1-3 settimane) di un componente del team per malattia

Azione #2: Per quel determinato periodo di tempo occorre distribuire il lavoro del componente malato ai membri con più disponibilità

* **Membro non collaborativo**: Presenza di un componente del team che non collabora e non rispetta le scadenze fissate dal Project Manager.

Azioni #6: Riunioni frequenti per discutere l’andamento del team.  
Se membro non collabora lo stesso, il team lo allontana per il bene del progetto

Rischi tecnologici (T):

* **Mancanza strumenti per il testing**: Rischio dovuto alla possibile mancanza di telefono con sistema operativo Android.   
  Attualmente due membri del team non possiedono dispositivi Android e non hanno la possibilità di testare l’app propriamente.   
  Il rischio è che almeno uno smartphone dei restanti tre membri si rompa e/o un componente di questi abbandoni il gruppo.

Azione #11: Ricerca di dispositivo per sopperire la mancanza.

* **Mancanza di strumentazione per lavoro/programmazione**: Rischio che un componente del gruppo non abbia possibilità di lavorare al progetto nei tempi prestabiliti per malfunzionamento dei propri strumenti (PC e rete).

Azione #8: il componente lavora da remoto e riferisce il lavoro ad un componente del gruppo che si occupa di aggiornare le informazioni online.

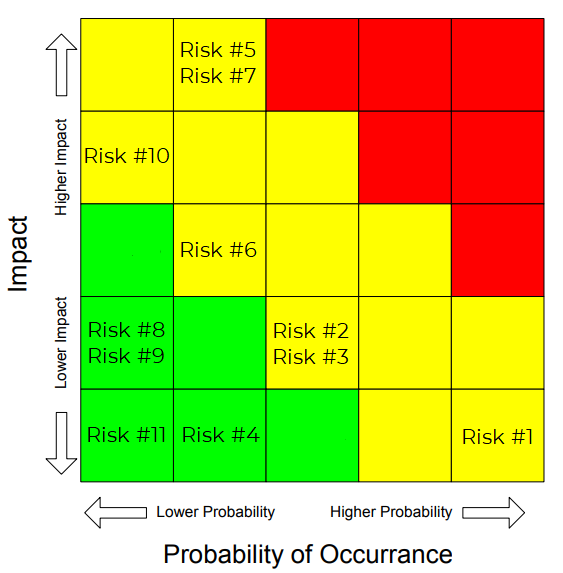
* **Perdita di dati:** Rischio di perdita dei dati relativi sia alla documentazione che al codice.

Azione #7: Eseguire frequentemente dei backup in locale e in un servizio di cloud

* **Strumentazione poco performante per qualche membro del team:** Nel caso un componente del gruppo possiede degli strumenti che non consentano una gestione rapida ed agevole del lavoro si rischia un rallentamento nel processo produttivo di questo componente.

Azioni #4: Prestare strumenti più performanti fino al termine del progetto; Il componente lavora da remoto e riferisce il lavoro ad un componente del gruppo che si occupa di aggiornare le informazioni online.

| **Rischio** | **Categoria** | **Probabilità** | **Impatto** | **RMMM** |
| --- | --- | --- | --- | --- |
| Incapacità di portare a termine bug fixes | R | 90% | 1 | #1 |
| Malattia membro del team | RU | 50% | 2 | #2 |
| Incapacità di portare a termine funzionalità secondarie | R | 50% | 2 | #3 |
| Strumentazione poco performante membro del team | T | 20% | 1 | #4 |
| Incapacità di portare a termine funzionalità principali | R | 15% | 5 | #5 |
| Membro non collaborativo | RU | 15% | 3 | #6 |
| Perdita dati | T | 15% | 5 | #7 |
| Mancanza strumentazione per lavoro/programmazione | T | 10% | 2 | #8 |
| Mancanza membro del team | RU | 5% | 2 | #9 |
| Snaturazione dell’idea originale di progetto dell’applicazione | R | 5% | 4 | #10 |
| Mancanza strumenti per il testing | T | 5% | 1 | #11 |



* 1. **Meccanismi di monitoraggio e di controllo**

Per comunicare verranno utilizzati i seguenti strumenti:

* Gruppo Whatsapp per le discussioni collettive
* Github per la condivisione del codice sorgente
* Google Drive per la condivisione ed il mantenimento dei documenti
* Trello e Jira per l’organizzazione del lavoro di gruppo
* Google Meet per dialogare e lavorare a distanza

I diversi aggiornamenti verranno erogati frequentemente e inoltre ci saranno riunioni periodiche (settimanalmente), sia a distanza che in presenza, con tutti i membri del team per valutare l’andamento dei vari task.

* 1. **Pianificazione dello staff**

Per sviluppare i task del progetto sono necessarie varie conoscenze, molte delle quali sono fornite durante lo svolgimento del corso di Ingegneria del Software, che comprendono:

- Tecniche e nozioni per l’organizzazione del progetto  
- Soft skills per la creazione dell’ambiente lavorativo  
- Programmazione in Android  
- Conoscenza del linguaggio XML  
- Programmazione in NodeJS (interfacciarsi con database NoSQL)

Sono richieste delle competenze che tutti i membri devono avere già acquisito durante i propri anni di esperienza, tra le quali:

- Programmazione in Java (Object Oriented Programming, detta OOP)  
- Conoscenza del funzionamento dei DBMS non relazionali e dei linguaggi   
 NoSQL (in particolare MongoDB Query Language, detto MQL)  
- Conoscenza del linguaggio di programmazione JavaScript

E’ necessario che tutti i membri abbiano almeno una conoscenza base degli elementi sopra descritti.   
Per i membri che si occupano di gestire il codice è necessaria una conoscenza ad un livello di dettaglio maggiore in modo da poter effettuare una corretta ed ampia revisione del codice prodotto da altri membri del team.

1. **Descrizione dei Processi Tecnici**

* 1. **Metodi, Strumenti e Tecniche**

In questo paragrafo verranno esposti le tecniche, i metodi e la strumentazione usata nei vari processi, è necessario definirli meticolosamente per uniformare il team ad adottare le medesime misure e strumenti nel progetto.

* **Ambienti di sviluppo:** la parte front-end del progetto sarà sviluppata tramite l’IDE Android Studio sviluppato da JetBrains, che permette inoltre di testare l'applicazione direttamente su un emulatore presente nel software stesso; Mentre per le modifiche e il settaggio dei parametri per la parte back-end verrà utilizzato l’editor testuale avanzato Visual Studio Code
* **Linguaggi:** lo sviluppo della business logic avverrà nel linguaggio Java, mentre per lo sviluppo delle interfacce grafiche si lavorerà con il linguaggio XML.

È però disponibile in Android Studio uno strumento di progettazione di interfacce grafiche che genererà automaticamente codice XML sulla base di drag and drop e modifiche di impostazioni.

Il salvataggio dei dati utente in locale avverrà in formato JSON per mantenere la persistenza di alcuni dati anche in assenza di connessione

Il linguaggio in cui è stato sviluppato il server dal team di Families Share è invece NodeJS, basato su JavaScript, e per il database viene utilizzato MongoDB Query Language (MQL)

* **Strumentazione di lavoro:** per lo sviluppo del progetto è necessario usare dei PC sistemi operativi Windows 10, 11, MacOS o Linux; mentre i requisiti tecnici minimi sono: 4 GB di RAM, processore o SoC da 1.5 gigahertz (GHz) o superiore, 32GB di spazio di archiviazione su disco e connessione ad internet
* **Strumentazione di test:** per effettuare le operazioni di testing verranno utilizzati smartphone sia fisici che sia virtuali, tramite l’emulatore messo a disposizione da Android Studio, dotati di sistema operativo Android 6.0+. Questi dispositivi però devono avere i seguenti requisiti minimi: 1 GB di RAM, 8GB di spazio di archiviazione, GPS integrato, connessione internet abilitata ed una versione Google Play Services 11.0+
* **Standards di sviluppo:** il codice sviluppato sarà strutturato in modo modulare e scalabile, sfruttando soprattutto i principi dell’incapsulamento. Questo ne permette una visualizzazione più leggibile e pulita, ed inoltre ne permette una maggiore manutenibilità sia da parte del team sia esterno, oltre che essere utile per un possibile ampliamento di funzionalità futuro (esempi in sezione 1.3).

In aggiunta, metodi, campi, variabili e commenti saranno scritti in inglese, la “lingua dell’informatica” sempre per un discorso di comprensibilità internazionale

* 1. **Documentazione del Software**

La documentazione del software sarà realizzata dal team durante lo sviluppo del progetto e sarà disponibile entro le scadenze indicate nella sezione 1.2. Inoltre, se sarà necessario, verranno apportate modifiche ai documenti in caso di rischedulazione dei compiti secondo la programmazione citata nella sezione 2.1.

* 1. **Funzionalità di supporto al progetto**

Come supporto per la realizzazione e l’avanzamento del progetto si sono adottate varie misure di comunicazione, essendo questo basato su lavoro di gruppo.

Qui vengono elencate le tecnologie e le piattaforme utilizzate:

* **Funzionalità di supporto**: il team utilizzerà GitHub come strumento di hosting, che sfrutta Git come strumento di versioning, ed abilita la memorizzazione distribuita e controllata di codice per agevolare lo sviluppo, coordinazione e progressione del progetto. Questa utility permette di mantenere una visione globale sul lavoro, anche da parte di professori e tutor, la cronologia commentata delle modifiche e la possibilità di rollback (ripristino) a versioni precedenti in caso di necessità.

Parallelamente, si utilizzerà la suite di Google, in particolare Google documents e Google Drive per mantenere e redigere la documentazione relativa al progetto. Si sono scelti questi strumenti perché offrono un'elevata versatilità e sono pensati per il team working anche a distanza, inoltre le modifiche sono salvate in cloud.

La comunicazione nel team invece avverrà principalmente servendosi del servizio di messaggistica WhatsApp

* **Pianificazione della qualità**: è necessario controllare lo sviluppo del codice mano a mano che vengono implementate features nell'applicazione. Questa pratica viene affidata ai membri del team che non le hanno sviluppate, aventi quindi una visione nuova e più oggettiva, o al soggetto identificato come backup engineer. Tale controllo verrà svolto rispettando determinati pattern di programmazione e la modularità del codice

1. **Pianificazione del lavoro, delle risorse umane e del budget**

* 1. **WBS (Work Breakdown Structure)**

Il progetto è strutturato secondo specifiche funzioni, attività e milestones, che insieme formano la work breakdown structure del progetto.

Le funzioni che sono state identificate permettono di monitorare costantemente l’avanzamento del progetto, anche nei dettagli dei vari processi, e forniscono inoltre la possibilità di assegnare e gestire le attività rispetto ai componenti del team.

Di seguito le **funzioni** identificate dal team:

1. **Project Management**

Funzione che sarà presente durante tutto lo svolgimento del progetto. Poiché viene adottato il modello di processo di tipo Agile, sarà necessario al team organizzare il lavoro per ogni settimana, mostrando il progresso e lo stato dei task assegnati. I problemi che potrebbero sorgere generalmente vengono risolti in maniera proattiva cercando di evitare ritardi e situazioni sconvenienti.

1. **Tecnologie software**

Le tecnologie con cui il team dovrà approcciarsi per sviluppare il progetto saranno Java, XML e NodeJS come linguaggi di programmazione; Android Studio e Visual Studio Code come ambienti di sviluppo. Come sistema per la persistenza dei dati il team utilizzerà MongoDB**.**

1. **Redazione dei documenti**

La redazione dei documenti avverrà prevalentemente durante la programmazione vera e propria dell'applicazione sia per mantenere un livello di organizzazione maggiore, sia per favorire una conoscenza comune tra tutti gli elementi del team con la possibilità di apportare modifiche ai vari documenti che potrebbe riportare dati non aggiornati.

* 1. **Piano di progetto**

Sviluppo e stesura dei requisiti per poter scrivere il piano di progetto.

* 1. **Documento di analisi e specifica**

Sviluppo e spiegazione dei vincoli e degli obiettivi riguardanti l’applicazione.

* 1. **Piano di testing**

Ideazione e realizzazione delle date per i relativi test per verificare l’operatività dell’applicazione.

* 1. **Documentazione progettazione**

Definizione dell’architettura del sistema, costruzione dei modelli UML, progettazione UI e UX.

1. **Controllo della qualità**

Controllo dell’applicazione, deve essere in linea con gli obiettivi descritti nel piano di progetto.

1. **Fase di test**

La fase di testing è la fase dove verranno effettuati dei test per verificare se lo sviluppo di features hanno portato ad un risultato e funzionamento atteso per il progetto finale dell’applicazione, oppure necessitano ancora tempo e lavoro per il completamento.

* 1. **Unit**

Test dei singoli task terminati

* 1. **Integration**

Test di tutti i task dopo aver integrato il task appena terminato.

Il progetto è stato diviso in più attività, alcune specifiche e alcune più generali, in questo caso sono state suddivise a loro volta in tasks.

**Attività** identificate dal team:

A. **Idea iniziale di progetto**

Tutte le attività effettuate dai membri del team per concordare una proposta iniziale del progetto del corso di Ingegneria del software.

B. **Analisi di fattibilità**

Vengono analizzati tutti gli obiettivi, i requisiti e i vincoli prendendo come punto di partenza l’idea progettuale del punto A.

C. **Bozza di documentazione**

In questa fase viene creata una bozza di documentazione per il progetto.

D. **Definizione dei ruoli**

Viene definita la tipologia di team utilizzata e di conseguenza i ruoli di ogni membro e le relative politiche/connessioni.

E. **Analisi dei requisiti**

In questa fase viene effettuata un’analisi dei requisiti tecnici decisi dal team.

F. **Analisi e gestione dei rischi**

Tramite delle attività di brainstorming del team vengono considerati tutti i possibili problemi che possono verificarsi durante la progettazione, lo sviluppo ed il testing dell’applicazione. Per ognuno di essi si definisce una possibile gestione.

G. **Creazione delle funzioni e delle attività**

Vengono create in questa fase tutte le attività e tutte le funzioni individuate dal team, con relative tempistiche.

H. **Costruzione WBS, Pert e Gantt**

Definizione dell’ordine in cui le varie attività verranno eseguite e la loro durata, tenendo conto dei vincoli temporali tra un’attività e l’altra.

I. **Progettazione della comunicazione client/server**

Sviluppo e test della comunicazione tra il server NodeJs e un’applicazione client Android.

I.1 **Creazione applicazione e configurazione dei parametri**

Sviluppo di un’applicazione Android basilare per effettuare test di comunicazione e configurazione corretta dei parametri lato server (NodeJs).

I.2 **Test della corretta comunicazione**

Test comunicazione client/server e comunicazione con il database (MongoDB).

J **Progettazione dell’applicazione Android**

Sviluppo dell’applicazione di base nella quale verranno implementate tutte le funzionalità principali del progetto.

J.1 **Creazione UI principale e Activities**

Sviluppo dell’interfaccia utente basilare per il progetto e creazione codice sorgente.

J.2 **Progettazione features principali e parte delle secondarie**

Progettazione generale delle classi e UI necessarie per realizzare le features principali (sezione 3.1)

J.3 **Collegamento UI-Activities**

Link fra l’interfaccia utente e le classi create precedentemente.

J.4 **Sviluppo delle Activities**

In questa fase vengono sviluppate le Activities per operare in funzione delle features scelte al punto J.2.

K **Perfezionamento dell’applicazione**

Sviluppo dell’applicazione Android finale con funzioni complete e aggiuntive.

K.1 **Miglioramento delle classi e Activities**

Controllo per migliorare la qualità generale del codice delle classi e delle Activities.

K.2 **Completamento progettazione delle funzionalità primarie**

Perfezionamento delle funzioni principali.

K.3 **Completamento progettazione delle funzionalità secondarie**

Perfezionamento delle funzioni secondarie.

L **Sviluppo UI definitiva**

Costruzione dell’interfaccia grafica definitiva user friendly ed in stile minimal

M **Features extra**

Possibile implementazione di chat dedicate, migliore integrazione con luoghi, orari di disponibilità e dark mode.

N **Consegna del progetto**

Test di controllo per il corretto funzionamento dell’applicazione, consegna della documentazione e del codice sorgente.

Per una corretta organizzazione e cadenza del progetto, il team ha individuato tre milestones che lo caratterizzano. Queste milestones saranno dei punti cardine e fondamentali del progetto, rappresentano importanti traguardi intermedi nello svolgimento del progetto

Queste sono le **milestones** identificate dal team:

Θ **Rilascio Demo**

Verrà rilasciata la versione dimostrativa dell’applicazione per mostrare il suo effettivo effettivo funzionamento nelle procedure di base, quali: il funzionamento della connessione con il server e il database esistenti, l’implementazione minimale delle funzionalità principali e di alcune delle secondarie (sezione 3.1)

Ω **Versione Completa**

Rilascio della versione completa dell’applicazione con ampliamento di alcune delle funzionalità principali e l’implementazione delle restanti funzionalità secondarie (sezione 3.1)

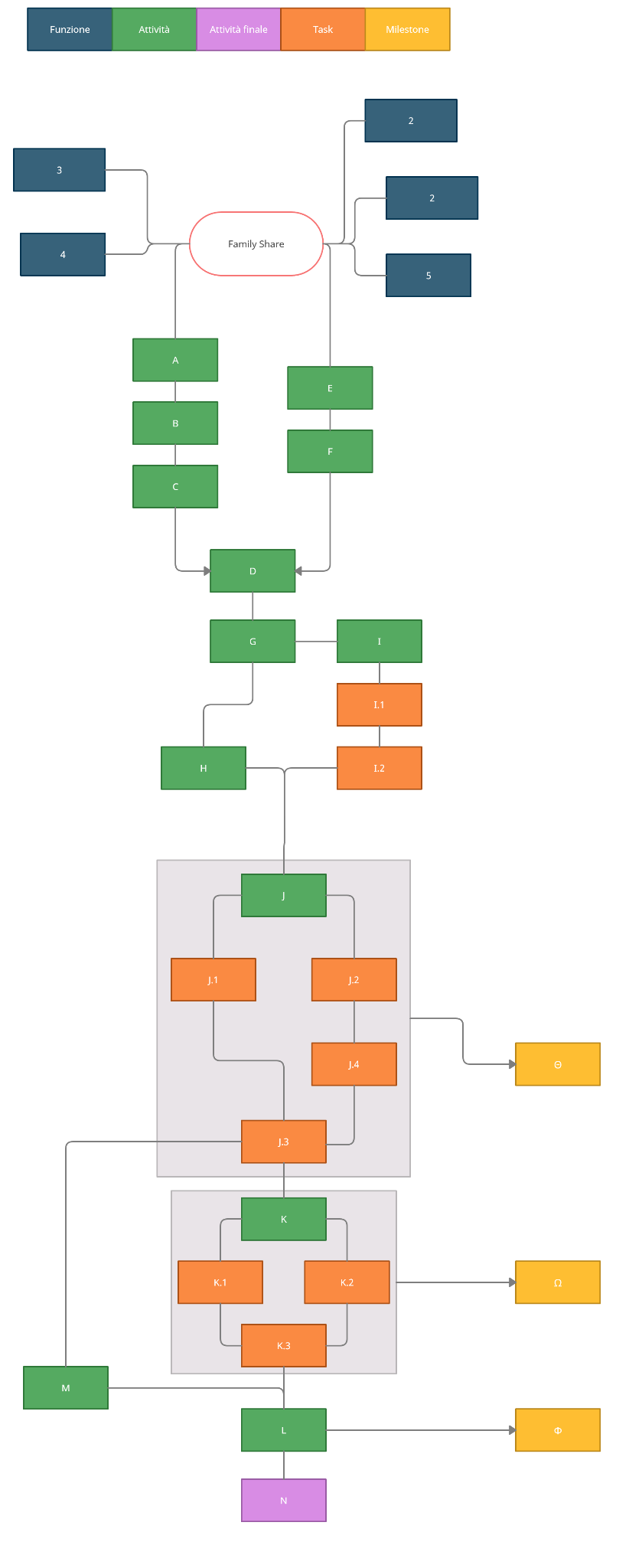
Φ **App finale**

Rilascio della versione finale dell’applicazione, con grafica definitiva e corretto funzionamento delle varie features. L’applicazione sarà quindi pronta per l’utilizzo da parte del pubblico.

Viene ora fornita una rappresentazione grafica della WBS, dove sono inserite tutte le funzioni, le attività, i task e le milestones secondo un ordine logico, in più si identifica anche un sequenzialità o parallelizzazione delle stesse attività (o task)

*Legenda:*



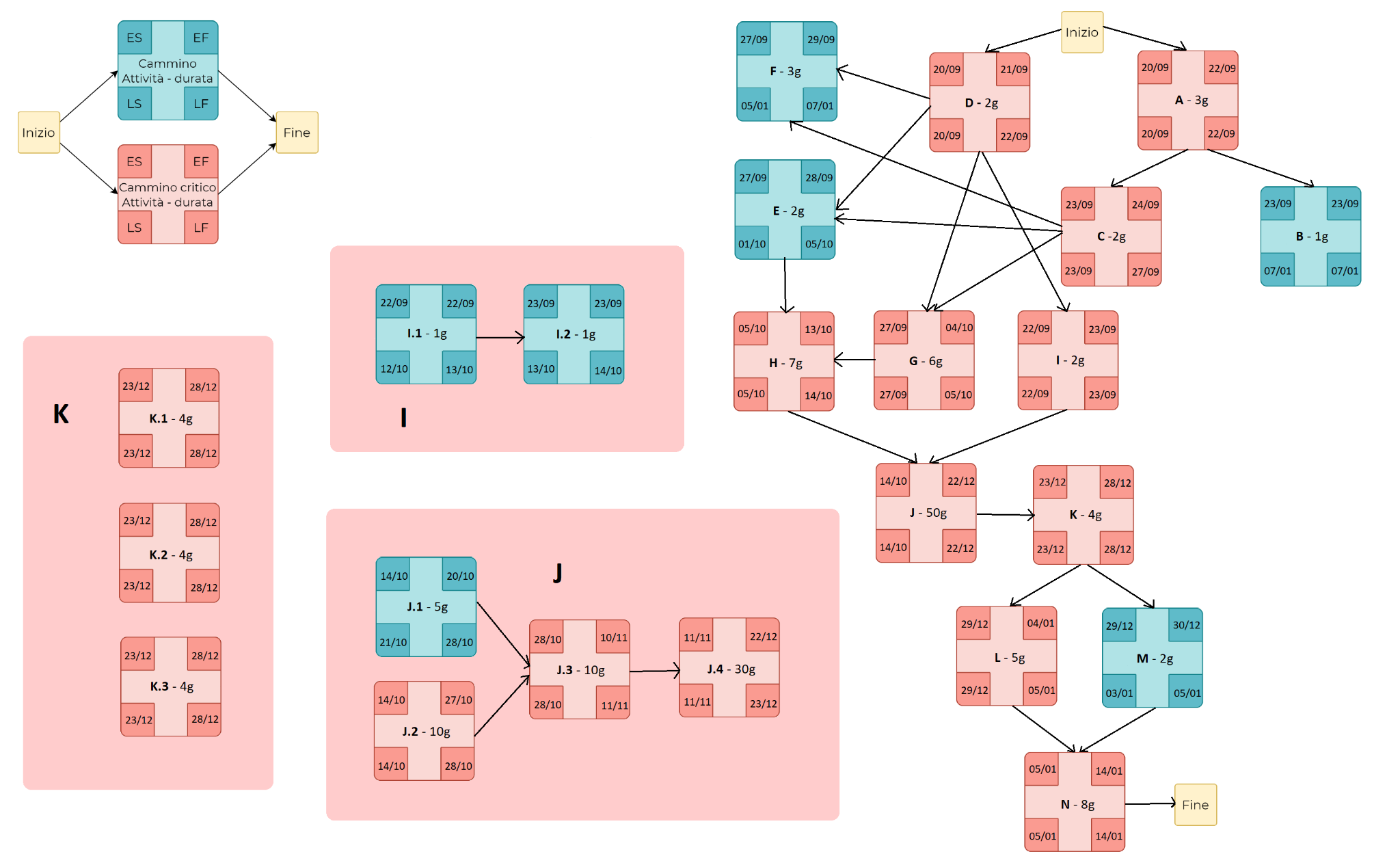


* 1. **Dipendenze**

Quando parliamo di dipendenze intendiamo tutti i vincoli presenti tra le varie attività di carattere temporale. Spesso, infatti, per iniziare un’attività è necessario che un’altra, ad essa propedeutica, sia terminata.  
  
Questa suddivisione è necessaria per osservare se alcune attività possano essere eseguite in parallelo, ottimizzando la loro realizzazione. Anche per questa ragione è stato deciso di adottare la metodologia Agile.  
  
Qui di seguito rappresentiamo la tabelle di tutte le attività pianificate insieme ai loro ES, EF, LS, LF (Vedi sezione 1.5 per maggiori dettagli)  
  
Il team e i vari componenti si impegnano a lavorare tutti i giorni dal lunedì fino al venerdì compreso, nelle ore libere in cui ciascun membro non ha lezione o lavoro.  
Vengono esclusi tutti i giorni festivi.  
  
Sulla base della tabella riportata in seguito è stato realizzato il Diagramma di Pert, in questo modo è possibile visualizzare con maggiore chiarezza tutti gli slot temporali necessari per ogni task e le varie esecuzioni che possono essere svolte in parallelo.

**Tabella delle dipendenze**

| **ID** | **Dipendenze** | **Durata** | **ES** | **EF** | **LS** | **LF** |
| --- | --- | --- | --- | --- | --- | --- |
| A | - | 3 gg | 20/09/21 | 22/09/21 | 20/09/21 | 22/09/21 |
| B | A | 1 gg | 23/09/21 | 23/09/21 | 07/01/22 | 07/01/22 |
| C | A | 2 gg | 23/09/21 | 24/09/21 | 23/09/21 | 27/09/21 |
| D | - | 2 gg | 20/09/21 | 21/09/21 | 20/09/21 | 22/09/21 |
| E | C - D | 2 gg | 27/09/21 | 28/09/21 | 01/10/21 | 05/10/21 |
| F | C -D | 3 gg | 27/09/21 | 29/09/21 | 05/01/22 | 07/01/22 |
| G | C - D | 6 gg | 27/09/21 | 04/10/21 | 27/09/21 | 05/10/21 |
| H | E - G | 7 gg | 05/10/21 | 13/10/21 | 05/10/21 | 14/10/21 |
| I | D | 2 gg | 22/09/21 | 23/09/21 | 22/09/21 | 23/09/21 |
| I.1 | D | 1 gg | 22/09/21 | 22/09/21 | 12/10/21 | 13/10/21 |
| I.2 | I.1 | 1 gg | 23/09/21 | 23/09/21 | 13/10/21 | 14/10/21 |
| J | H - I | 50 gg | 14/10/21 | 22/12/21 | 14/10/21 | 22/12/21 |
| J.1 | I | 5 gg | 14/10/21 | 20/10/21 | 21/10/21 | 28/10/21 |
| J.2 | I | 10 gg | 14/10/21 | 27/10/21 | 14/10/21 | 28/10/21 |
| J.3 | J.1 - J.2 | 10 gg | 28/10/21 | 10/11/21 | 28/10/21 | 11/11/21 |
| J.4 | J.3 | 30 gg | 11/11/21 | 22/12/21 | 11/11/21 | 23/12/21 |
| K | J | 4 gg | 23/12/21 | 28/12/21 | 23/12/21 | 28/12/21 |
| K.1 | J | 4 gg | 23/12/21 | 28/12/21 | 23/12/21 | 29/12/21 |
| K.2 | J | 4 gg | 23/12/21 | 28/12/21 | 23/12/21 | 29/12/21 |
| K.3 | J | 4 gg | 23/12/21 | 28/12/21 | 23/12/21 | 29/12/21 |
| L | K | 5 gg | 29/12/21 | 04/01/22 | 29/12/21 | 05/01/22 |
| M | K | 2 gg | 29/12/21 | 30/12/21 | 03/01/22 | 05/01/22 |
| N | L - M | 8 gg | 05/01/22 | 14/01/22 | 05/01/22 | 14/01/22 |



* 1. **Risorse necessarie**

Sono necessarie diverse categorie di risorse per portare a termine il progetto:  
  
**Risorse umane**: Insieme di tutti i componenti del team comprese le loro conoscenze in tutti gli ambiti necessari (descritti nelle sezioni precedenti)  
  
**Risorse hardware**: Tutti gli strumenti tecnologici necessari per la realizzazione dell’obiettivo; consiste nei PC e nei dispositivi Android   
  
**Risorse software**: Sono necessari tutti gli ambienti di sviluppo come Android Studio e un IDE per sviluppare NodeJS. E’ inoltre richiesto l’utilizzo di Google Drive, Whatsapp e Google Meet per comunicare (connessione internet necessaria)  
  
**Risorse temporali**: Consiste nel tempo necessario per la realizzazione del progetto. Ogni componente deve avere la capacità di gestire il proprio tempo e sapersi coordinare con le altre attività che deve svolgere (come, ad esempio, studio, lavoro e sport)  
  
**Risorse documentative**: E’ necessaria tutta la documentazione relativa alla programmazione Android, JavaScript, NodeJS, MongoDB.  
E’ di fondamentale importanza anche la documentazione relativa all’applicazione esistente: Family Share

* 1. **Allocazione del Budget e delle risorse**

Il progetto non presenta alcun costo aggiuntivo per il team poiché tutti gli strumenti necessari sono con licenza gratuita, o sono dispositivi già in possesso dei vari membri

E’ possibile stimare la spesa totale per la realizzazione del progetto in base al quantitativo di ore di lavoro di ciascun membro del team: un totale di 10 ore di manodopera la settimana per ciascuno.  
Considerando complessivamente 11 settimane di lavoro e un pagamento medio di 12 euro l’ora otteniamo un quantitativo di denaro equivalente a 1320 euro per persona.

Il costo complessivo stimato: 6.600 euro.

Nota:

Nella stima è stato considerato solamente il tempo di lavoro in gruppo.

Il conteggio delle ore di lavoro individuale di ciascun membro è affidato alla responsabilità dello stesso, vista la scelta di operare con una metodologia Agile.

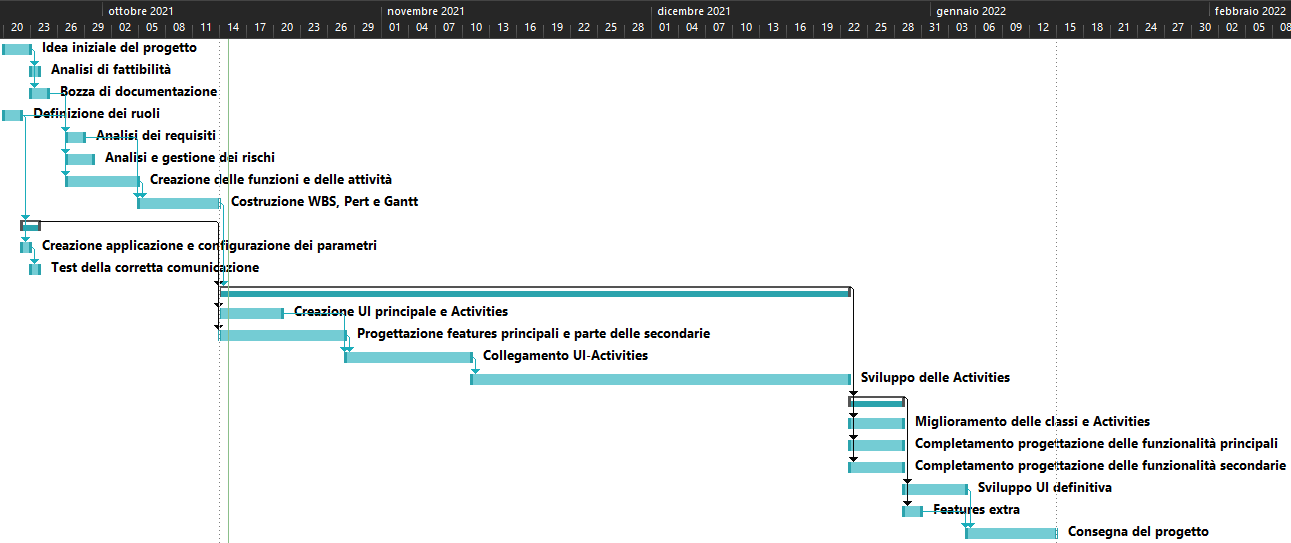
| Gruppo | Settore | Prezzo | Totale Parziale |
| --- | --- | --- | --- |
| Ore contate | Uomo-ora \* 5 | € 600 | € 6.600 |
| Documenti | Piano di progetto | € 250 | € 1.350 |
| Documento di analisi e specifica | € 500 |
| Piano di testing | € 300 |
| Documento di progettazione | € 300 |
| Features | Tracciamento Contagi | € 1.500 | € 3.250 |
| Bacheca | € 500 |
| Mercatino digitale | € 500 |
| Funzionalità secondarie \* 4 | € 600 |
| Bug Fixes \* 3 | € 150 |
|  |  | Totale finale (IVA esclusa) | € 11.200 |

* 1. **Pianificazione**

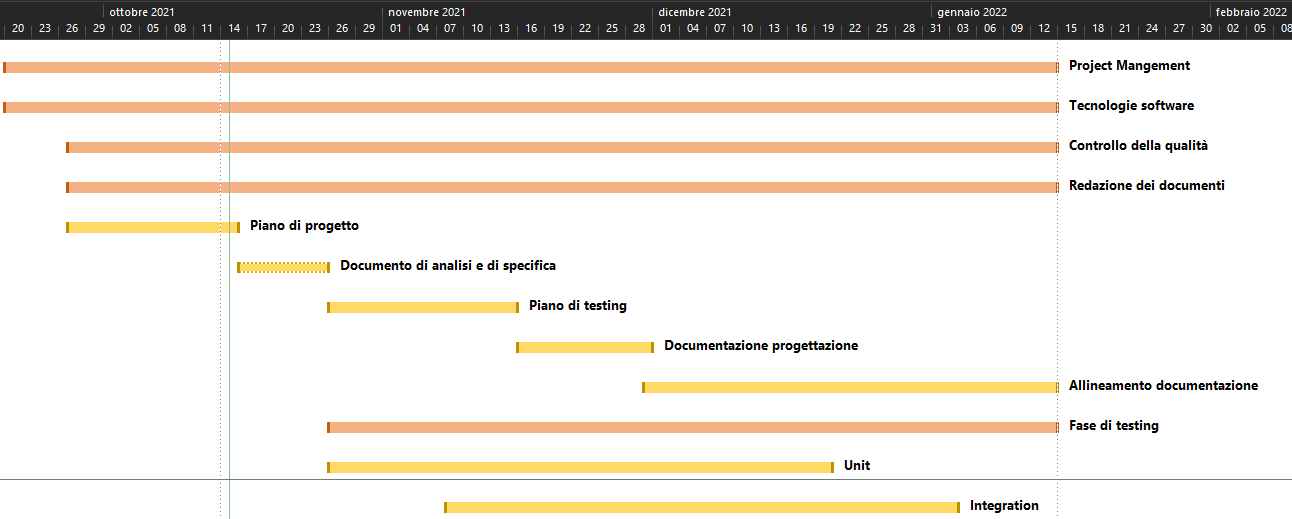
Durante lo sviluppo del progetto sono state fissate alcune deadlines, necessarie per mantenere sempre un ritmo di lavoro adeguato. E’ necessario non completare e consegnare un documento oltre la data prefissata nella deadline poichè causerebbe una propagazione di ritardi difficilmente gestibile.  
  
Di seguito sono riportate le Deadlines riguardanti la consegna della documentazione:

* Proposta iniziale: 29/9/2021;
* Piano di Progetto: 15/10/2021;
* Documento dei Requisiti: 25/10/2021;
* Piano di testing: 14/11/2021;
* Milestone 1: Rilascio beta (versione 1.0 codice sorgente): 15/12/2021;
* Milestone 2: Versione completa: 29/11/2021;
* Documento di Progettazione: 30/11/2021;
* Milestone 3: App finale (versione 1.1. del codice) e allineamento documentazione: 15/01/2022;

**Diagramma di Gantt**

****

**Funzioni**



1. **Riferimenti**

Per creare questo documento sono stati utilizzati come riferimento:

* “Piano di Progetto” di alcuni gruppi degli anni precedenti
* materiale messo a disposizione dal professore