121/01/2022







Corso

[CT0090] INGEGNERIA DEL SOFTWARE (CT3) - a.a. 2021- 22

Team  
****

Membri

ANDREA BRION 860595

FABIO DANESIN 882805

MARCO TAMISARI 865233  
TANJIN HABIBUR 875737

TOMMASO GOLFETTO 876451



**PIANO DI PROGETTO**

VERSIONE 1.2



**INDICE**

1. **INTRODUZIONE**
   1. **Overview**
   2. **Deliverables**
   3. **Evoluzione**
   4. **Glossario - Materiale di riferimento**
   5. **Glossario - Definizioni ed abbreviazioni**
2. **ORGANIZZAZIONE DEL PROGETTO**
   1. **Modello**
   2. **Struttura organizzativa**
   3. **Interfacce**
   4. **Responsabilità**
3. **DESCRIZIONE DEI PROCESSI GESTIONALI**
   1. **Obiettivi e priorità**
   2. **Vincoli, dipendenze e assunzioni**
   3. **Gestione dei rischi**
   4. **Monitoraggio e controllo**
   5. **Pianificazione dello staff**
4. **DESCRIZIONE DEI PROCESSI TECNICI**
   1. **Metodi, strumenti e tecniche**
   2. **Documentazione SW**
   3. **Funzionalità di supporto al progetto**
5. **PIANIFICAZIONE** 
   1. **Wbs**
   2. **Dipendenze**
   3. **Risorse**
   4. **Allocazione budget**
   5. **Pianificazione**
6. **RIFERIMENTI**



1. INTRODUZIONE

1.1 OVERVIEW

Usando un dispositivo Android o un’emulazione di esso, è richiesto lo

svolgimento delle fasi di: Analisi, Progettazione, Sviluppo e Testing.

Attraverso l’unità computazionale sopra-indicata implementeremo funzionalità aggiuntive all’applicazione “Families share”.

Di seguito, le istruzioni sull’uso dell’applicazione:

<https://www.families-share-toolkit.eu/wp-content/uploads/2020/11/App-slide-mod-ita.pdf>

L’applicazione è stata sviluppata da un consorzio di: università, centri di ricerca e organizzazioni non-profit d’Europa, impegnato nella progettazione di pratiche innovative di conciliazione famiglia e lavoro basate sul time-sharing.

La finalità è quella di fornire un supporto alle famiglie attraverso attività doposcuola, gruppi di interazione, servizi di assistenza, consulenza genitoriale e così a seguire.

Questa soluzione è nata in risposta alle nuove situazioni sociali createsi causa pandemia Covid, l’idea è di portare nuovamente dei momenti di tranquillità, crescita e gioco in una realtà dove sono molte le limitazioni e preoccupazioni date dal virus e dalle sue conseguenze sociali e sanitarie.

Destinata sia alle famiglie con difficoltà ad organizzare il tempo ludico e non dei figli, sia per coloro che vogliono passare del tempo assieme ai propri e a legare con altre famiglie.

Si parlerà più approfonditamente dell’applicazione nella sezione 3.1.

Alleghiamo, per concludere, il nostro github, dove condividiamo il codice sorgente ed il lavoro svolto periodicamente:

<https://github.com/FabioDanesin/Ingegneria-del-software-2021>

1.2 DELIVERABLES DEL PROGETTO

Le scadenze relative al progetto sono le seguenti:

* 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 (15/01/2022)
* Allineamento documenti (25/01/2022)



1.3 EVOLUZIONE DEL PROGETTO

Sfruttando la modularità e scalabilità del codice già esistente, andremo a

sviluppare i seguenti punti:

* Implementazione di servizi relativi alle statistiche Covid.
* Inserimento mappa durante la creazione di un’attività.
* Segnalazione positività/negatività ai gruppi a cui si partecipa in modo

che gli utenti contattati possano organizzarsi per mettersi in  
quarantena/sicurezza.

1.4 MATERIALI DI RIFERIMENTO

Per il progetto il team si servirà dei materiali didattici presenti sulla

piattaforma Moodle del corso.

Per la gestione del database il team ha scelto di attingere da:

<https://docs.mongodb.com>

Per la prototipazione il team ha scelto di utilizzare il seguente strumento: <https://www.figma.com>

Per formarsi nel linguaggio e tecnologie dell’applicazione verranno utilizzate le seguenti piattaforme:

* [Youtube](https://www.youtube.com)
* [Udemy](https://www.udemy.com)
* Siti di documentazione ufficiali
* Lezioni del corso CT0090 - Ingegneria del software

1.5 GLOSSARIO

* **Google drive**: Servizio web di memorizzazione e sincronizzazione online introdotto da Google nel 24 aprile 2021.

Fonte: <https://it.wikipedia.org/wiki/Google_Drive>

* **NodeJS**: Runtime system open source multipiattaforma orientato agli eventi per l’esecuzione di codice JavaScript.

Fonte: <https://it.wikipedia.org/wiki/Node.js>

* **Visual Studio Code**: Editor di codice sorgente.

Fonte: <https://it.wikipedia.org/wiki/Visual_Studio_Code>

* **MongoDB**: DBMS (Database Management System) non relazionale, orientato ai documenti.

Fonte: <https://it.wikipedia.org/wiki/MongoDB>

* **Webstorm**: Ambiente di sviluppo integrato per JavaScript e tecnologie correlate.
* **PostMan**: Postman è una piattaforma API per la creazione e l'utilizzo di API. Postman semplifica ogni fase del ciclo di vita dell'API e ottimizza la collaborazione in modo da poter creare API migliori, più velocemente.
* **Discord** : Piattaforma di comunicazione tra utenti dove ci si può messaggiare, chiamare, condividere file o fare streaming.
* **Draw IO**: Software di disegno grafico multipiattaforma gratuito e open source sviluppato in HTML5 e JavaScript.
* **API**: Acronimo di Application Programming Interface, serve ad indicare un gruppo di procedure già disponibili al programmatore per la realizzazione di un determinato compito.
* **Telegram**: Applicazione di comunicazione tra utenti dove si può messaggiare o condividere file.
* **React**: Libreria open-source, front-end, JavaScript per la

creazione di interfacce utente. È mantenuto da Facebook e da una

comunità di singoli sviluppatori e aziende.

Fonte: <https://it.wikipedia.org/wiki/React_(web_framework)>.

* **App**: Applicazioni, quelle per dispositivi mobili, vengono identificate

semplicemente come App.

Fonte: <https://it.wikipedia.org/wiki/Applicazione_(informatica)>.

* **Android**: Android è un sistema operativo per dispositivi mobili sviluppato da Google, progettato principalmente per sistemi embedded quali smartphone e tablet, con interfacce utente specializzate per televisori, automobili, orologi da polso, occhiali, e altri.
* **Branch**: un branch è un ramo. Un ramo è un insieme univoco di

modifiche. Ogni repository può avere più branch.

* **Master branch**: Branch principale del progetto, creato quando si fa

l’upload del primo file nella propria repository.

* **JavaScript :** è un linguaggio di programmazione orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione web lato client(esteso poi anche a lato server) per la creazione di siti web e applicazioni web.

Fonte: <https://it.wikipedia.org/wiki/JavaScript>

* **Pull request**: Richieste che consentono di comunicare agli altri le

modifiche che sono state inviate a un ramo di una repository su GitHub. Aperta una richiesta pull, è possibile rivedere le modifiche con i collaboratori e aggiungere commit di follow-up prima che le modifiche vengano unite al master branch.

* **GitHub**: Servizio di hosting di repository che fornisce un’interfaccia

grafica basata sul web, il controllo dell'accesso e diverse funzionalità

per la collaborazione.

* **Hosting**: Un servizio di rete che consiste nell'allocare su un server online delle pagine web di un sito/applicazione web, rendendolo così

accessibile dalla rete Internet e ai suoi utenti.

Fonte: <https://it.wikipedia.org/wiki/Hosting>

* **Host**: Termine relativo, il significato pratico dipende dal contesto. In

breve, l'host è il "corrispondente" da raggiungere o cui si è collegati.

Fonte: <https://it.wikipedia.org/wiki/Host>.

* **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)**: Indica il minimo giorno in cui l’attività può

terminare dato Es e la durata dell’attività.

* **LF (latest finish time)**: Indica il giorno massimo in cui l’attività deve finire in modo da non creare ritardo alle attività che dipendono da essa.
* **LS (latest start time)**: Indica il giorno massimo in cui l’attività deve

iniziare senza provocare ritardi alle attività che dipendono da lui dato LF e la durata dell’attività.

* **Scrum**: Indica il processo iterativo della metodologia Agile.
* **User stories**: Descrizioni semplici di requisiti e funzionalità indicate

dall’utente.

* **Product backlog**: Elenco di caratteristiche, requisiti, funzionalità di cui

necessità il prodotto.

* **Sprint backlog**: E’ un product backlog da usare durante uno sprint (nel

metodo classico si usa una WBS).

* **Task**: Indica il lavoro svolto dal team per completare un articolo del

product backlog.

* **Daily scrum**: Riunione rigorosamente in piedi quotidiana della durata di massimo 5/10 minuti.
* **Sprint**: Intervallo di tempo, dalla durata di meno di 30 giorni, nella quale viene creato un incremento di prodotto.
* **Incremento di prodotto (MVP)**: Ciò che si vuole produrre al termine di

ogni sprint, detto anche Minimo prodotto funzionante.

Deve essere funzionante, usabile e fattibile.

Esempio: Sviluppando un intero sito web, un MVP potrebbe essere

una o più pagine del sito.



2. ORGANIZZAZIONE DEL

PROGETTO

2.1 MODELLO DEL PROCESSO

Il team si organizza in un modello Agile con tipologia SCRUM.

Modello che è caratterizzato da un ciclo di eventi che permette al team di

centrare meglio lo scope del progetto e lavorare in modo più “rilassato”,

a scapito di un uso maggiore di tempo e risorse.

Questo modello viene detto un modello iterativo, cioè vi è una procedura

predefinita che si divide in:

* **Backlog refinement**: Evento dove il product owner discute delle user stories col team, su cui poi egli pone la priorità.

Il team invece aiuterà a calcolare le stime dei costi e risorse necessarie

per ciascuna (Raffinamento detto Grooming).

Viene inoltre definito il concetto **done**, cioè un’insieme di caratteristiche per considerare la user story finita.

* **Sprint planning**: Evento dove presenziano tutte le entità di scrum, dove il product owner espone le user stories con relative priorità nel backlog e il team sceglie cosa sviluppare.

Queste consisteranno nel sprint backlog.

* **Daily scrum**: Riunione breve (max 15 minuti) dove scrum master e team si aggiornano sullo stato dei lavori si chiedono
  + Cosa è stato fatto ieri
  + Cosa farò oggi
  + Quali sono gli impedimenti di ieri e come superarli

E’ importante che non sia usata per problem solving.

Da fare ogni volta che il team si incontra per lavorare.

* **Sprint review**: Riunione informale tenuta al termine di ogni sprint, dove il team consegna l’incremento (che ovviamente rispetta il concetto **done**).
* **Sprint retrospective**: Terminato lo sprint si identificano le buone e cattive abitudini del team rivedendo i propri processi con il fine di migliorarli, il product owner non partecipa.

Per essere più efficienti il team si organizza con meeting settimanali per

identificare i compiti da svolgere, suddividerli e fare una ricapitolazione di ciò che è stato fatto in quella settimana.

Verranno organizzate le attività considerando un margine per gli imprevisti e un ordine da rispettare nella sequenza di operazioni da svolgere con eventuali sovrapposizioni, anticipazioni o posticipazioni.

Gli incontri settimanali avverranno secondo la seguente programmazione:

| **Giorno** | **Martedì** | **Mercoledì** | **Giovedì** | **Venerdì** |
| --- | --- | --- | --- | --- |
| **Fascia oraria** | **Sera** | **Pomeriggio** | **Sera** | **Sera** |
| **Descrizione** | **Incontro su Discord/Zoom** | **Incontro per trattare problematiche riscontrate la settimana precedente** | **Incontro su Discord/Zoom** | **Incontro su Discord/Zoom** |
| **Durata** | **60-120 min** | **120 min** | **60-120 min** | **60 min** |

Useremo una scrum board semplificata che divide le attività da fare in: To-do, Doing, Done, dove metteremo rispettivamente attività da svolgere, in

svolgimento e terminate relative alla settimana/sprint in corso.

In questo modello non è detto che tutte le attività vengano svolte, se al

termine di uno sprint non si è completato un’attività il product owner può

decidere se tenerla, inserendola nel prossimo sprint o nel backlog con una

data priorità oppure di scartare l’idea.

Le attività saranno identificate e assegnate direttamente ad un componente

del team con le attività presenti nella sezione 5.1.

Per capire l’iter di sviluppo seguirà una breve descrizione della metodologia

Agile: il product owner riceve le richieste dal cliente, comunica col team l’analisi di tale richieste e crea un backlog contenente queste divise per priorità e definisce i criteri d’accettazione (**done**).

Lo scrum master si assicura un ambiente di lavoro consono alla metodologia.

Il team crea i deliverables necessari e mostra i progressi al product owner

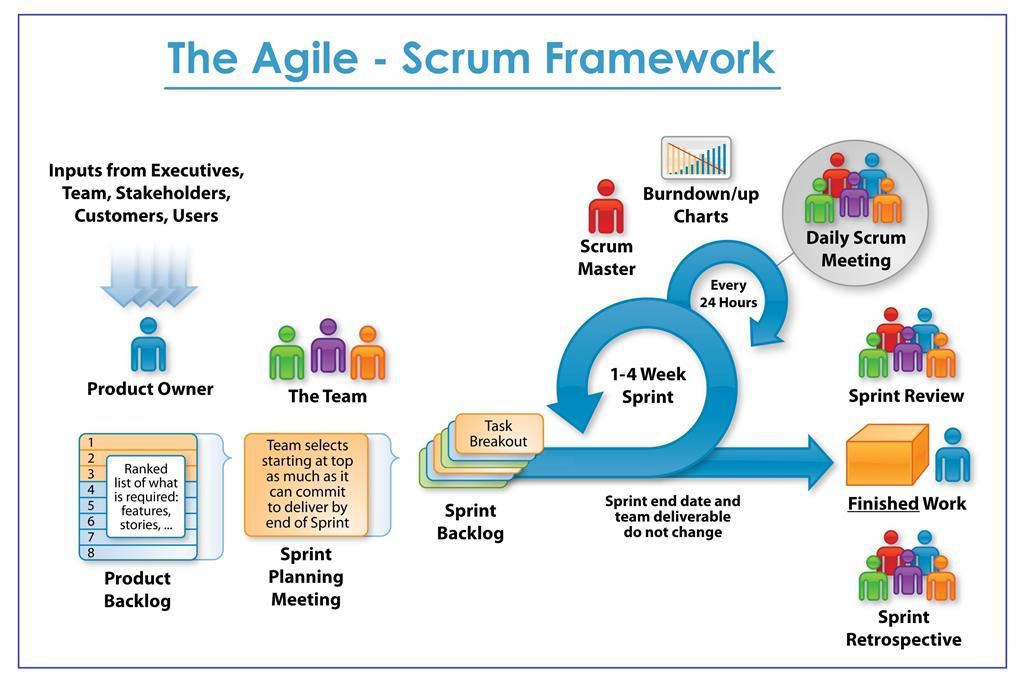
durante i review meeting.

Il product owner consegna valore attraverso il rilascio dell’incremento.

Ogni giorno di lavoro vi è una riunione di 15 minuti chiamata daily scrum.

Al termine di ogni sprint team e product owner si trovano per discutere di

possibili miglioramenti nei prossimi sprint.

****

Le fasi principali del progetto sono:

* Analisi e definizione del progetto;
* Progettazione del sistema e del software
* Creazione di una demo dell’app;
* Implementazione e test delle singole componenti;
* Integrazione e test del sistema;
* Sviluppo della versione completa dell’app;
* Installazione e mantenimento.

Ogni fase sarà caratterizzata da una durata, da un certo margine d’errore e

verrà posta in relazione con le altre per la schedulazione (con sovrapposizioni, anticipazioni e posticipazioni).

2.2 STRUTTURA ORGANIZZATIVA

Per quanto riguarda la struttura organizzativa si è scelto un’organizzazione a progetto, dove serviranno sottostrutture che si occupano di

amministrazione, risorse umane ecc.

La divisione di responsabilità però non è verticale riguardo le funzioni, bensì ogni funzione del progetto ha una persona che ne è responsabile.

Abbiamo scelto questo principalmente per l’ottimo livello di coordinazione e

comunicazione all’interno del team, la flessibilità decisionale e l’estrema

velocità e determinazione nel rispondere alle esigenze degli stakeholder a

scapito di un uso delle risorse efficiente e di un rischio di esplosione dei costi nel caso il progetto duri più del previsto.



Inoltre definendo un Projectmanager

possiamo ottenere sia un maggiore focus sullo svolgimento, sia un controllo maggiore sullo stato dei lavori del team in modo da ridurre i ritardi delle consegne.

Egli ha completa visibilità e controllo sul budget del progetto, sulle risorse e sulle assegnazioni del lavoro alle risorse.

Infine questa struttura permette l’approvazione di piani e obiettivi associati dai vertici della struttura.

La tipologia di team scelta è quella Democratica-Decentralizzata.

In questa non c’è un leader permanente, bensì le decisioni vengono prese solo all’unanimità, in quanto essendo gli elementi direttamente coinvolti nel

processo l’interesse di ciascuno riguarda l’andamento dell’attività.

La comunicazione è detta orizzontale in quanto tutto il team è allo stesso

livello,infatti non c’è un leader riconosciuto che coordina il lavoro.

Fra i vantaggi di questa tipologia c’è che ognuno ha un’attitudine positiva a

ricercare errori nello sviluppo.

I contro riguardano principalmente che non è scalabile e che non è imponibile al team se non vi sono le condizioni corrette.

2.3 INTERFACCE ORGANIZZATIVE

Durante la realizzazione del progetto, il team dovrà interfacciarsi con persone esterne al team.

In particolare, le entità sono:

* Monitoraggio: Riguardo al monitoraggio e feedback, il team ha come punto di riferimento il professor Agostino Cortesi.

Vi sono 2 elementi del team che si occuperanno di ciò: Tommaso Golfetto, Andrea Brion.

* Consultazione: Riguardo la consultazione nelle fasi di sviluppo, il team farà riferimento al professor Agostino Cortesi e alla tutor Alessia Michela Di campi.

I membri del team che si occuperanno di ciò sono: Fabio Danesin, Tommaso Golfetto e Tanjin Habibur.

* Stakeholder: Riguardo l’approccio con gli stakeholder (coloro che possono essere influenzati o influenzare le decisioni, attività o risultati del progetto) il team si affiderà a: Marco Tamisari, Fabio Danesin e Tanjin Habibur.

2.4 RESPONSABILITA’ DEL PROGETTO

Per concludere questo secondo capitolo vediamo i vari ruoli dei membri del

team con compiti e responsabilità.

I compiti sono stati assegnati stando ai punti di forza dei vari membri, senza

compartimentare troppo il lavoro, in modo così che tutto il team abbia una

visione generale dei compiti altrui.

Approccio utilizzato per ridurre al minimo i rischi riguardanti assenze dei

membri durante lo sviluppo.

* Product owner: E’ il responsabile del product backlog, quindi lo tiene

aggiornato, ordinato e visibile a tutti.

Egli è il responsabile di massimizzare il valore del prodotto e del lavoro del team e del contatto col cliente.

Svolto da Tommaso Golfetto

* Scrum master: E’ colui che si assicura che la metodologia sia compresa ed usata correttamente da Team e Product owner, deve far capire ad esempio come gestire gli elementi del product backlog, o comprendere la pianificazione del prodotto.

E’ colui che massimizza il valore creato dal team, tramite assistenza e

attività di coaching.

Svolto da Andrea Brion

* Team: Entità che si occupa dello sviluppo dell’incremento, è strutturata e autorizzata ad organizzare il proprio lavoro, infatti:
  + Nessuno indica come debbano trasformare le user stories in incrementi (auto-organizzati);
  + Hanno varie competenze (cross-funzionali);
  + Non contengono sotto-team.

Team composto da: Tommaso Golfetto, Tanjin Habibur, Fabio Danesin, Marco Tamisari.



3. DESCRIZIONE DEI PROCESSI GESTIONALI

3.1 OBIETTIVI E PRIORITA’

Il team lavorerà con il linguaggio Javascript.

I principali obiettivi sono implementare lo sviluppo di funzionalità aggiuntive all’applicazione già esistente (vedi 1.3)

Inoltre, è previsto il rilascio di una demo (versione beta) e di documentazione dettagliata del lavoro svolto (il team si prefigge una propria scadenza).

Fra le funzionalità, il team ha scelto di poter segnalare il contagio all’interno

dei gruppi di cui si fa parte, in modo tale da avvisare tutti i membri e agire di

conseguenza.

Un altro esempio di ciò che il team propone è poter visualizzare statistiche

riguardanti la situazione Covid o anche la possibilità di vedere il luogo su Google Maps dove si terrà un evento creato da un utente.

3.2 VINCOLI, DIPENDENZE E ASSUNZIONI

E’ ovviamente molto importante frequentare le lezioni in quanto ottime per apprendere conoscenze velocemente e per confrontarsi con le varie entità associate (professori e tutor), nel caso per vari motivi si mancasse ad una di queste, conviene, quanto prima, aggiornarsi riguardo le tematiche trattate dai colleghi.

Lo sviluppo di un progetto richiede delle nozioni tecniche, come conoscere:

● Il linguaggio di programmazione (JavaScript);

● La struttura dei database NoSQL;

● L’ambiente di sviluppo NodeJs;

● Servizio di hosting GitHub.

NodeJs e Android verranno affrontati in diverse lezioni frontali al posto di lezioni del corso, per le restanti invece il team dovrà attingere ed imparare dal web e dagli strumenti dati dal corso come pdf esplicativi riguardo l’utilizzo ed il funzionamento dei vari sopracitati.

Chiaramente vi sono delle assunzioni di una buona condotta professionale, quali l’essere diligenti, responsabili e pronti ad aiutare i colleghi in modo da conseguire ciascun task nel modo più efficiente e completo possibile.

Infine non possiamo dimenticare che strumenti, materiali e mezzi utili e

utilizzati per portare a termine il lavoro settimanale ed il progetto stesso sono subito accessibili, non avendo quindi ritardi di alcun tipo.

3.1 GESTIONE DEI RISCHI

La strategia di gestione dei rischi scelta dal team è la strategia preventiva.

In questa tipologia si identificano i rischi potenziali, valutandone probabilità

ed impatto (effetti) stabilendo poi un ordine di priorità.

Il piano costruito permetterà di reagire più velocemente ed in modo efficace ai rischi, inoltre il team otterrà benefici maggiori gestendo rischi “grandi”.

Di seguito, i rischi rilevati dal team:



Di seguito la tabella da leggere come una matrice: (Probabilità x Impatto)



3.4 MECCANISMI DI MONITORAGGIO E CONTROLLO

Come canali di comunicazione principali verranno utilizzati Telegram e

Discord, per la condivisione del codice sorgente e schedulazione del lavoro la piattaforma Github, e per la documentazione, Google Drive.

Sono inoltre previsti degli incontri fissi infrasettimanali per confrontarsi sul lavoro svolto e discutere di eventuali problematiche da risolvere. Verranno inoltre proposte nuove idee e pensieri sull’andamento del progetto e i prossimi task da eseguire.

La suddivisione dei compiti e i relativi controlli fanno riferimento al punto 2.4 del presente documento.

3.5 PIANIFICAZIONE DELLO STAFF

Per sviluppare funzionalità dell’applicazione in questione sono necessarie delle nozioni pregresse, come la conoscenza JavaScript, la struttura e funzionamento base dei database NoSQL, conoscenze base di GitHub.

Alcune conoscenze sono state conseguite affrontando vari esami degli anni

precedenti come basi di dati, in cui abbiamo imparato a gestire database ed altre invece che sono extracurricolari e quindi andranno fissate nel proprio tempo libero (come Node Js e React).

Terminata l’app sarà compito del team analizzarla e determinarne la realizzazione.

Se valutata positivamente si passerà allo sviluppo del punto successivo, se no si rivedranno i punti appena affrontati.



4. DESCRIZIONE DEI PROCESSI TECNICI

4.1 METODI, STRUMENTI E TECNICHE

In questo paragrafo esporremo i processi tecnici e la strumentazione utilizzata nei processi.

E’ necessario definirli in modo da facilitare il team ad adottare uguali misure

all’interno del progetto.

● Ambiente di sviluppo: Utilizzeremo gli editor di codice sorgente Visual Studio Code e Webstorm. Saranno utili per tutti i membri del team poiché sono dotati di un supporto per debugging e un controllo per Git integrato.

Per progettare i grafici nella WBS del piano di progetto, i membri addetti alla documentazione utilizzeranno Draw IO.

● Linguaggi: Il progetto verrà sviluppato utilizzando il linguaggio di

programmazione JavaScript, utilizzato nella programmazione Web lato client (React), e anche al lato server (Node Js), per la creazione di effetti dinamici interattivi tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulle pagine web o applicazioni web.

● Strumentazione di lavoro: Per lo sviluppo serviranno dei computer dotati di sistema operativo Windows 10 (con i seguenti requisiti tecnici minimi: 8 GB di RAM, 64 GB di spazio libero su disco) e con relativa connessione di rete.

● Per il testing verranno utilizzati diversi browser (Chrome e Firefox).

● Standards di sviluppo: Riguardo al codice è necessario strutturare e

modellare il codice affinché sia modulare e scalabile.

Questo modo permette una visualizzazione più pulita e facile da leggere: viene così semplificato il lavoro all’interno del team per maneggiare sezioni di codice scritte da altri membri del team o comunque per sfruttare il meccanismo di incapsulamento.

Inoltre, questo permetterà, di testare le singole componenti ed effettuare

manutenzioni, se necessario, su di essi. Inoltre è obbligatorio l’utilizzo dei

commenti per spiegare ad altri membri del team, i quali si occuperanno di

controllare la qualità ed il corretto il funzionamento delle componenti appena sviluppate, la qualità del codice stesso.

Inoltre, è necessario questo requisito per permettere ai membri di questo

progetto di facilitare l’evoluzione del progetto (descritta nella sezione 1.3).

4.2 DOCUMENTAZIONE DEL SOFTWARE

Come citato nella sezione 1.2, la documentazione del software verrà

implementata durante le varie scadenze del progetto.

Il team si impegnerà ad apportare eventuali modifiche ai vari documenti

costantemente, in modo tale da tenere aggiornato l’avanzamento del lavoro

con ordine e precisione.

4.3 FUNZIONALITA’ DI SUPPORTO DEL PROGETTO

Essendo un progetto di gruppo, la comunicazione è fondamentale per il

successo del progetto.

Queste verranno viste come strumenti di supporto per il progetto.

Elenchiamo qui sotto le principali funzionalità di supporto adottate del team:

● Pianificazione della qualità: Il controllo periodico del codice durante lo sviluppo e implementazioni di funzionalità è fondamentale. Esso infatti verrà eseguito dal team in modo da essere più accorti ed avere più pareri riguardo eventuali problematiche legate al codice.

Il controllo verrà eseguito considerando la modularità del codice stesso ed eseguendo frequentemente dei test, in modo da rendere la probabilità di svista di una complessità irrisoria.

● Pianificazione della gestione delle configurazioni: ogni membro del team lavorerà su un proprio branch, ogni versione del codice sarà scalabile e visualizzabile in qualsiasi momento.

● Funzionalità di supporto: il team utilizzerà la piattaforma GitHub per

mantenere il codice sorgente con i relativi backup, dove, sia il team che il responsabile di tutorato potranno controllare l'avanzamento del progetto, è molto utile inoltre per tornare a versioni precedenti (possibilmente funzionanti) in caso di malfunzionamenti importanti sul software che si sta sviluppando attualmente.

Parallelamente, si utilizzerà Google Drive per mantenere e redigere la

documentazione relativa al progetto.



5. PIANIFICAZIONE DEL LAVORO, DELLE RISORSE UMANE E DEL BUDGET

5.1 WORK BREAKDOWN STRUCTURE

La **WBS (Work Breakdown Structure)** serve per identificare i compiti (attività) e funzioni del progetto. Dividendo il lavoro in sottoattività, è possibile avere un maggior controllo di tutto il progetto e facilitarne la realizzazione.

Vengono rilevate delle **attività**, ovvero le unità principali di lavoro, che vengono a loro volta suddivise in **task**.

Le **milestones** rappresentano i punti finali di ogni singola attività di processo, mentre i **deliverables** i risultati forniti al committente.

I task sono unità di lavoro “atomiche”, con durata stimabile che necessitano di certe risorse: producono dei risultati tangibili (documentazione, codice, ...). La fase di scheduling di un progetto prevede quindi di dividere il carico di lavoro in attività e mansioni, stimando tempo e risorse necessarie per portarle a termine. Occorre inoltre tenere in considerazione eventuali rischi che potrebbero rallentare il processo.

Di seguito, le **attività** (1 - 12) rilevate dal team.

1 **Ideazione**

Comprende le attività di brainstorming tra i membri del team per

ideare un possibile progetto da sviluppare per il corso di Ingegneria del software.

2 **Analisi Families Share**

E’ la fase che prevede lo studio e l’analisi del progetto “Families Share”:

è fondamentale comprendere le varie funzionalità dell’applicazione

per ideare e avviare il nostro lavoro.

3 **Analisi di fattibilità**

Questa attività prevede di apprendere le varie tecnologie

dell’applicazione, in particolare Javascript, e analizzare gli

obiettivi, requisiti e vincoli per realizzare il progetto.

4 **Analisi rischi**

Previsione di eventuali rischi durante la realizzazione del progetto

e strategie da applicare nel caso si verificassero.

5 **Ruoli team**

Definizione dei ruoli e compiti per ogni membro del team.

6 **Meeting**

Fase che prevede di stabilire e fissare dei meeting con il team ogni

settimana: elenco lavoro svolto, discussione di eventuali problematiche

e soluzioni, prossimi task da eseguire.

7 **Scadenze e Documentazione**

Analisi delle varie scadenze da rispettare, quali:

Piano di Progetto, Documento di analisi e specifica, Piano di testing,

Documento di progettazione.

8 **Stesura WBS**

In questa fase il progetto viene suddiviso in attività e funzioni: è

utile ad aver un maggior controllo del carico di lavoro e avere (tramite grafici) una previsione del tempo impiegato a realizzarlo.

9 **Prototipazione grafica**

Sviluppo di un’interfaccia grafica tramite lo sviluppo di un mockup.

10 **Progettazione Demo**

Attività più sostanziosa, che viene a sua volta divisa in tasks, per

facilitarne lo sviluppo ed il controllo.

10.1 **Running applicazione**

Fase primordiale della progettazione, è necessario

configurare l’applicazione affinché si connetta ai vari servizi .

10.2 **Preparazione ambiente di sviluppo e Github**

Questo task prevede la preparazione dell’ambiente di sviluppo e delle varie tecnologie, nel nostro caso Visual Studio Code, per iniziare a programmare (linguaggio Javascript).

Inoltre, tutti i membri del team lavoreranno con la piattaforma di condivisione Github.

10.3  **Interfacciamento database**

Abbiamo creato un DB(database) locale e abbiamo lavorato su quello

10.4 **Programmazione Funzionalità**

Sviluppo delle funzionalità scelte durante la prima fase del

progetto, consiste nel programmarle e testarle periodicamente affinché soddisfino i requisiti descritti nel piano di progetto.

11 **Progettazione Finale**

Al susseguirsi del rilascio della demo, vi saranno ulteriori

aggiustamenti e un arricchimento di funzionalità minori con lo scopo

di proporre una versione finale migliore.

12 **Consegna del progetto**

Ultimi test dell’applicazione e successiva consegna della

documentazione completa e del codice sorgente.

Queste sono le **funzioni** (A - D) rilevate dal team.

A **Project Management**

Funzione attiva durante tutto il corso del progetto, che permette una

migliore organizzazione e suddivisione delle attività. Verrà adottata

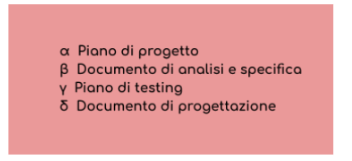
la metodologia Agile, come indicato nel punto 2.1.

B **Documentazione**

Durante lo sviluppo del codice sorgente, i documenti ci permetteranno

di tracciare qualsiasi attività elencandone tutti i vari principi e dettagli.

Prevede lo sviluppo di 4 documenti:



C **Controllo qualità**

L’evoluzione dell’applicazione verrà controllata in modo costante. Le

attività hanno un tempo stabilito per essere eseguite e questa

funzione controllerà la validità tale.

D **Testing**

Funzione in cui tutte le implementazioni verranno testate per

verificarne il corretto funzionamento, fase utile per capire se ci sia

bisogno di modifiche o aggiustamenti.

Il team ha rilevato 2 milestones (★), importanti traguardi dello svolgimento del progetto.

★ **Rilascio versione beta**

Obiettivo del team è progettare e rilasciare una versione beta con le

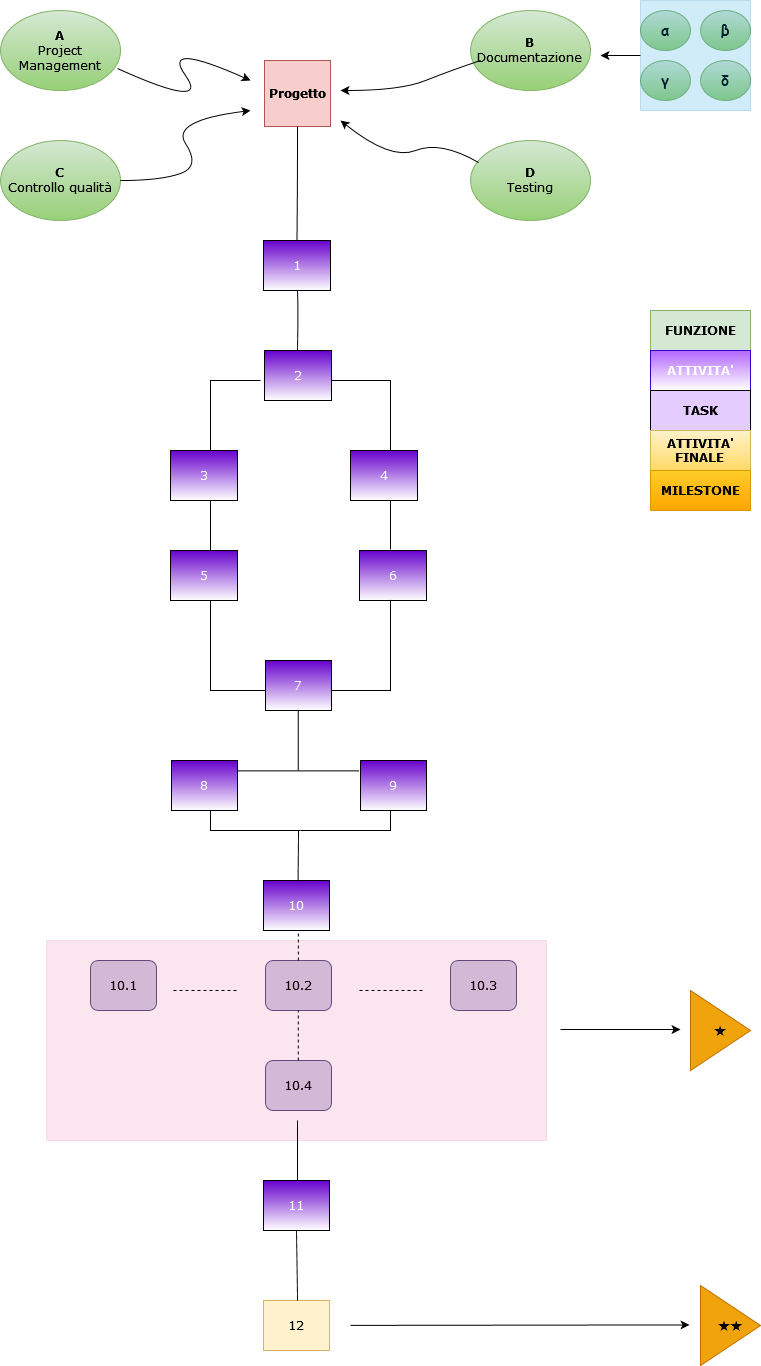
funzionalità principali, accompagnate da documentazione che ne espone i dettagli.

★★ **Applicazione finale**

Consiste nel rilascio dell’applicazione finale, con le funzionalità

implementate in modo dettagliato come visto nel piano di progetto.

Il seguente grafico illustrerà e riassumerà visivamente ciò che è stato descritto in questo paragrafo della Work Breakdown Structure.



5.2 DIPENDENZE

Si ha una dipendenza quando l’inizio o la fine di un’attività dipende dall’inizio o dalla fine di un’altra. Una semplice regola consiste nel verificare che ogni attività abbia un predecessore, a meno che sia collegata alla data di inizio del progetto, e un successore a meno che sia l’ultima attività del progetto.

La prima dipendenza rilevata è tra la seconda attività e la terza: prima di

poter eseguire l’analisi di fattibilità (e anche quella dei rischi poiché possono

essere eseguite nello stesso momento) è necessario analizzare il progetto

datoci in modo tale da comprenderne le funzionalità e dettagli.

Un’altra dipendenza è quella di apprendere e programmare le varie scadenze da settembre a gennaio e cercare di rispettarle al meglio, altrimenti sarebbe impossibile compiere l’ottava attività, ovvero la stesura della Work Breakdown Structure.

Le ultime attività sono quelle più sostanziose e cariche di lavoro, per questo

motivo verranno svolte una alla volta, la decima con i propri task determinano la prima milestone, cioè il rilascio di una versione beta del progetto.

Il lavoro termina con l’undicesima e dodicesima attività, che riguardano gli

ultimi test ed il rilascio dell’applicazione (seconda milestone).

Il team lavorerà anche nei giorni del weekend, poiché durante la settimana

alcuni membri lavorano o si dedicano allo studio di altre materie. Sono previsti dei giorni di pausa, come per esempio quelli di Natale e Capodanno. Di seguito è riportata una tabella delle dipendenze per visualizzare le attività e la stima del tempo per impiegarle.

Sono stati dati giorni con ampio margine alle attività più consistenti, poiché la funzione della documentazione ricoprirà abbastanza tempo durante tutto lo svolgimento del progetto.

Nella tabella, le sigle ES, EF, LS e LF significano rispettivamente:

**- Early Start,**

**- Early Finish,**

**- Late Start,**

**- Late Finish.**

TABELLA DIPENDENZE

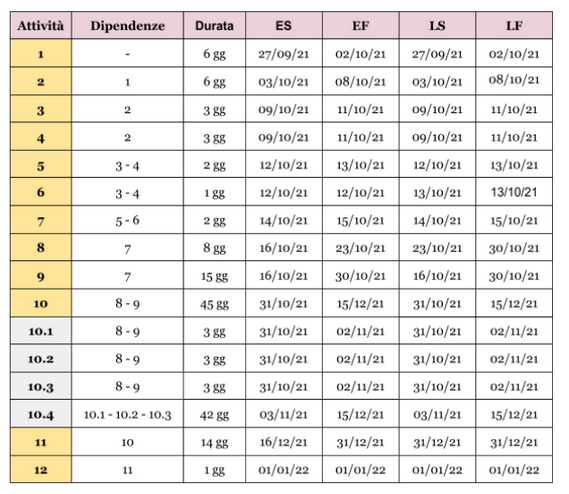


DIAGRAMMA DI PERT



5.3 RISORSE NECESSARIE

Esistono molteplici risorse necessarie per lo svolgimento e conseguimento del progetto, vediamole:

● Risorse umane: Date dai membri del team e dalle loro conoscenze ed

abilità.

● Risorse hardware: Strumentazione necessaria (laptop, dispositivi test

ecc).

● Risorse software: Software necessari come NodeJs, Google drive, Git,

Telegram, VisualStudio code, MongoDB, WebStorm, PostMan, Discord.

● Risorse temporali: Ore lavorative impiegate dai membri.

● Risorse di documentazione: Documentazione necessaria per sviluppare l’applicazione, come quella per comprendere come funzionano le varie API usate e i requisiti di progetto.

5.4 ALLOCAZIONE DEL BUDGET E DELLE RISORSE

Lo sviluppo del progetto non prevede spese economiche, in quanto l’ambiente di utilizzo è gratuito (Discord, Visual Studio Code, Draw IO, Google Drive, Github).

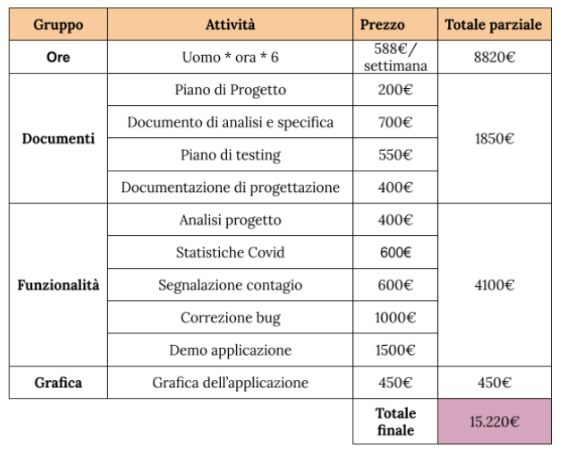
La tabella sottostante riassume la quantità di lavoro svolta con il relativo

prezzo, considerando i vari incontri fissi settimanali (sezione 2.1) e supponendo che ogni membro del team venga pagato 15 Euro ogni ora, il calcolo è il seguente:

7h ore settimanali \* 6 membri \* 15€ = 588€

588€ è il prezzo settimanale, calcolando l’inizio dello svolgimento del progetto al 27 settembre e la fine il 2 gennaio, vi sono circa 14 settimane.

588€ \* 14 = 8820€ **(costo base stimato)**.



5.5 PIANIFICAZIONE

Come citato nei paragrafi precedenti, vi sono delle scadenze riguardanti la

documentazione del progetto, e oltre a queste vanno aggiunte le milestone

rilevate.

● **Piano di progetto** (15 ottobre 2021)

● **Documento di analisi e specifica** (25 ottobre 2021)

● **Piano di testing** (14 novembre 2021)

● **Documentazione di progettazione** (30 novembre 2021)

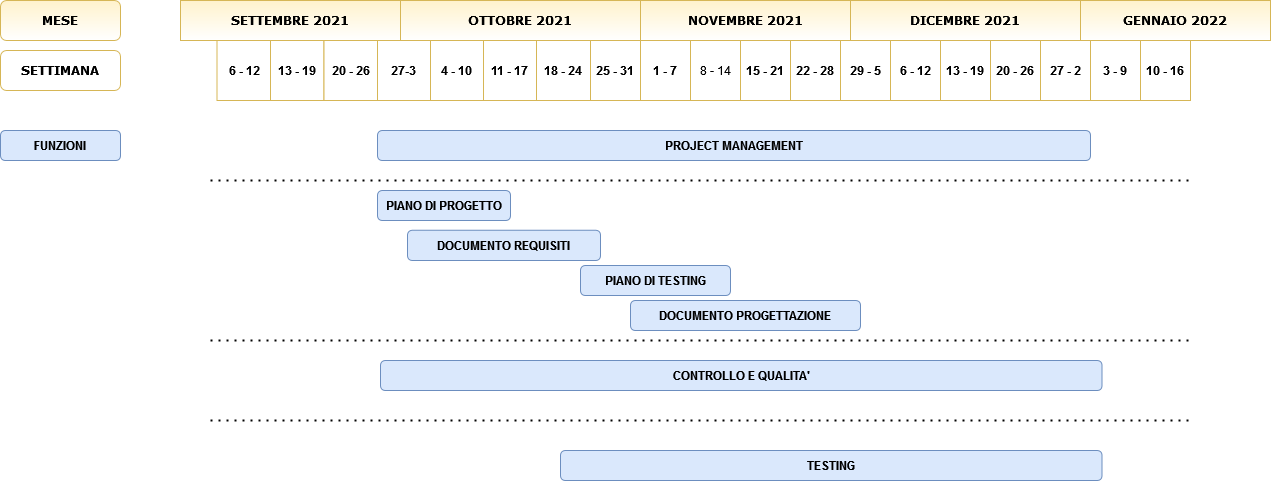
● **Milestone 1: Rilascio demo - Versione 1.0 codice sorgente** (15 dicembre

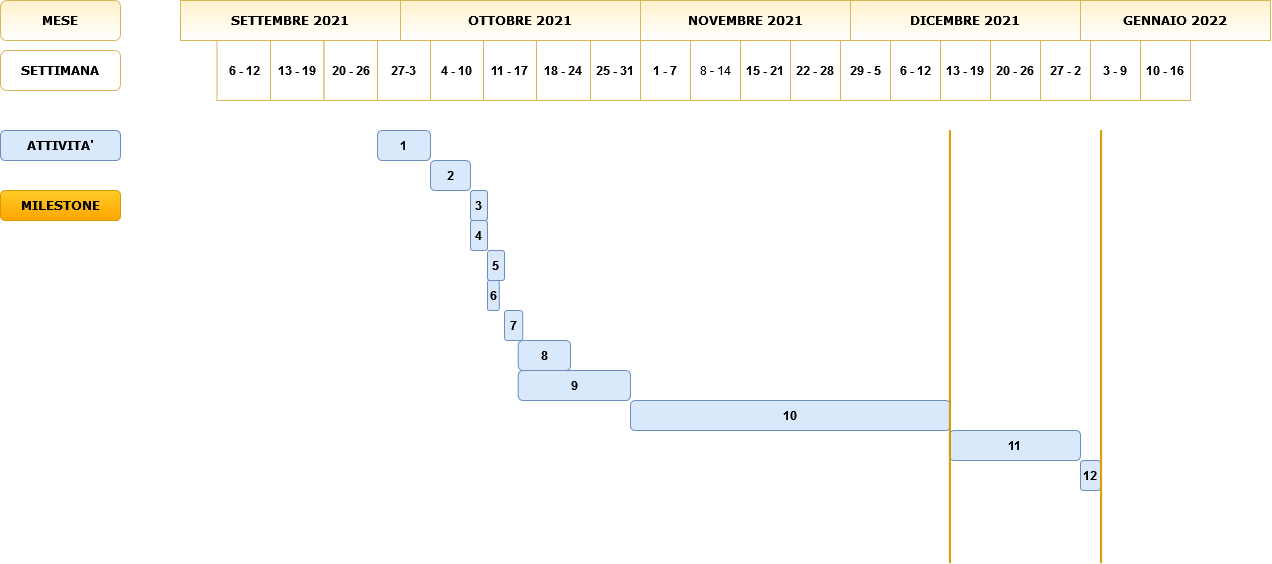
2021)

● **Milestone 2: Applicazione Finale** (2 gennaio 2022)

● **Versione 1.1 codice sorgente** (15 gennaio 2022)

DIAGRAMMI DI GANTT







6. RIFERIMENTI

Per la realizzazione di questo documento sono stati utilizzati come riferimento:

* Documenti di alcuni gruppi degli anni precedenti.
* Materiale messo a disposizione del professore.
* Materiale ricercato in rete.

Per la realizzazione di questo documento sono state impiegate:

* Tommaso Golfetto (30h)
* Andrea Brion (17h)
* Tanjin Habibur (7h)
* Marco Tamisari (2h)

