

Project Plan

Gabriele Chignoli

Maggio 2025

Contents

1	Introduzione	2
2	Modello di Processo	2
3	Organizzazione	2
4	Standard, Linee Guida, Procedure	2
4.1	Convenzioni di Nomenclatura	2
4.2	Documentazione	3
4.2.1	Versioning	3
5	Attività di Gestione	4
6	Rischi	4
7	Personale	4
7.1	Membri	4
8	Metodi e Tecniche	4
8.1	Requisiti	5
9	Garanzia di Qualità	5
10	Pacchetti di Lavoro	5
11	Risorse e Budget	5
12	Cambiamenti	6
13	Consegna	6

1 Introduzione

PantryManager è un applicativo che si propone di gestire l'inventario alimentare nell'ambito domestico. Un software che permetta di tener traccia di diversi dati riguardanti prodotti alimentari, aiutando l'utente a ridurre lo spreco, consumare prodotti sempre freschi, produrre pietanze più varie e mantenere quindi anche una dieta più equilibrata.

Il team ha intenzione di produrre un software che renda la gestione dell'aspetto alimentare della vita dell'utente meno onerosa, occupandosi di tenere traccia della scorta delle vivande in possesso, per poi aiutare a costruire un piano alimentare giornaliero basato sui prodotti a disposizione. Si vuole rendere inoltre tale processo il più personalizzabile possibile, fornendo così all'utente più autonomo dal punto di vista alimentare la possibilità di gestire in modo accurato la propria dieta.

2 Modello di Processo

Il team si impegna a seguire un modello di processo principalmente tradizionale, basato sul modello a cascata, che utilizza l'aspetto "pesante" di tale processo per avere una visione generale del software che mantenga lo sviluppo entro binari ben definiti. Si ritiene tuttavia necessario ridurre la rigidità del processo, utilizzando anche tecniche di tipo evolutivo, per garantire l'evoluzione e la soddisfazione dei requisiti ad ogni passo del processo, soprattutto durante le fasi di design, implementazione e testing. L'argomento viene approfondito nella sezione 1, *Ciclo di Vita del Software*, del documento *Gestione del Progetto*.

3 Organizzazione

Il progetto non entrerà in contatto con l'effettivo cliente a cui è destinato, né nella fase di elicitazione e validazione dei requisiti, né in fasi di test del software effettivo. L'unico tipo di feedback presente sarà in caso quello richiesto ai professori e professoresse del corso di Ingegneria del Software dell'Università degli studi di Bergamo, che forniranno in caso istruzioni di modifica per il miglioramento dell'applicativo e della sua documentazione.

4 Standard, Linee Guida, Procedure

4.1 Convenzioni di Nomenclatura

Vengono elencati di seguito gli standard adottati per la scrittura del codice (Java)

Componente	Convenzione
Classi	<i>Pascal Case</i> - i nomi iniziano con una lettera maiuscola. Se il nome contiene più parole, tutte le parole iniziano con una lettera maiuscola.
Metodi	<i>Camel Case</i> - i nomi iniziano con una lettera minuscola. Se il nome contiene più parole, tutte le parole iniziano con una lettera maiuscola.

Variabili	<i>Snake Case</i> - i nomi iniziano con una lettera minuscola. Se il nome contiene più parole, tutte le parole iniziano con una lettera minuscola e son separate da "_".
Costanti	<i>Full Uppercase</i> - tutte le lettere del nome sono maiuscole. Nel caso il nome fosse composto da più parole, queste si separano da "_".
Variabili Enum	<i>Full Uppercase</i> - tutte le lettere del nome sono maiuscole. Nel caso il nome fosse composto da più parole, queste si separano da "_".

Codice di Esempio

```
public class ClasseDiEsempio{
    int variabile_di_esempio = 0;
    static final int COSTANTE_DI_ESEMPIO;

    public static void metodoDiEsempio(){
        System.out.println("Hello World!");
    }
}
```

4.2 Documentazione

La documentazione verrà continuamente aggiornata e verificata, sia dal punto di vista della struttura che dei contenuti, per far sì che i documenti siano in linea con l'applicativo che si sta costruendo. I documenti verranno prodotti in Latex, e caricati in formato PDF su *GitHub*; verranno caricati ogni qualvolta vengono eseguite modifiche rilevanti (non si prevede che vengano eseguiti commit per la correzione di semplici errori grammaticali). Quando i documenti saranno completi, verrà attribuita ad essi una versione, e verrà pubblicata un issue su *GitHub* per l'accettazione. Verrà tenuta la cronologia delle versioni direttamente nel repository.

4.2.1 Versioning

Per tenere traccia dei documenti PDF prodotti verrà utilizzata una pratica basata sul *Semantic Versioning*, secondo la quale il nome di ogni file verrà seguito da un codice di tre cifre, separate da punti, che indicano rispettivamente (da sinistra verso destra) il grado della modifica operata sul documento, che può essere:

- **Major** - aggiunta/rimozione di grandi sezioni, accettazione di parte del documento come completa...
- **Minor** - aggiunta/rimozione di moderate parti di testo/diagrammi, riscrittura sezioni...
- **Patch** - correzioni di ortografiche/di terminologia, piccoli cambiamenti grafici/di struttura...

Esempio di nomenclatura file

Specifica_dei_requisiti 0.2.5

Si intende inoltre denotare i documenti ancora in continua revisione, e quindi non ancora nella prima versione accettata come finale, con la prima cifra impostata a 0 (zero).

5 Attività di Gestione

Il team si propone di stendere settimanalmente un breve report per riassumere il lavoro terminato durante i 7 giorni, le difficoltà riscontrate e i prospetti per la settimana seguente. Questi documenti non sono da considerarsi vincolanti rispetto all'effettivo avanzamento dello sviluppo, ma vengono utilizzati per tener traccia del lavoro svolto, per una possibile analisi post completamento, e in caso, per enti esterni interessati al processo, fornendo così anche un lavoro di manutenzione preventiva. Sostituiscono in parte la funzione di una *Stand-up meeting* del metodo *Agile*.

6 Rischi

La principale preoccupazione riguarda la gestione del tempo e l'effettiva capacità di fornire una versione funzionante del software, con i requisiti fondamentali soddisfatti, entro la data di consegna del prodotto.

Si prevede inoltre la possibilità dell'elicitazione di un eccessivo numero di requisiti (*campelli e fischietti*), che, pur potendo portare del valore aggiunto all'applicazione, richiederebbero troppe risorse per essere implementate entro i limiti di consegna.

7 Personale

Il team è composto attualmente da un solo membro, tuttavia verrà simulata la presenza di un secondo sviluppatore per sfruttare alcune funzionalità di *Github* e operare in modo più "realistico". Le principali aree di competenza richieste per lo sviluppo sono:







- Sviluppo e design dell'interfaccia grafica (Front-end Developer)
- Sviluppo e design dell'architettura
- Sviluppo e gestione della base di dati (Back-end Developer)
- Testing

7.1 Membri

- Gabriele Chignoli [Mat. 1073781]

8 Metodi e Tecniche

Il progetto prevede l'utilizzo di diverse tecnologie:

- [GitHub](#)  per il controllo della versione
- [Latex](#) (su piattaforma [Overleaf](#) ) per la produzione della documentazione
- [Eclipse](#)  come IDE per la scrittura del codice, con il quale verranno utilizzati i plug-in
 - [JUnit](#)  come framework per il testing
 - [Stan4j](#) per l'analisi statica del progetto
 - [CodeMR](#)  per l'analisi statica del progetto
- [Eclipse Papyrus](#)  per la produzione dei diagrammi UML (Universal Modelling Language)

- Il codice sarà scritto in [Java](#), con il quale verranno utilizzate le tecnologie
 - [Apache Maven](#) per la gestione del progetto e delle sue dipendenze
 - [H2](#) per la creazione e gestione di un database SQL
 - [Log4j](#) per avere funzionalità di logging più avanzate
 - [Hibernate](#) ORM (*Object-Relational Mapping*) per mappare gli oggetti Java in tabelle relazionali
 - [JPA](#) per interagire con il database *H2* senza utilizzare SQL nativo

8.1 Requisiti

- [Windows 10/11*](#)
- [JDK 21**](#) (or newer)

**Il software non verrà prodotto e testato per funzionare su sistemi operativi diversi da Windows 10/11; non si assicura dunque, pur essendo sviluppato in Java, il funzionamento su macchine con sistema operativo MacOS, Linux, Unix, etc.*

***L'applicazione per essere eseguita richiede l'installazione di un JRE (Java Runtime Environment), che tuttavia non viene più incluso all'interno del download di Java dal sito ufficiale. Per questo si consiglia di scaricare direttamente il JDK (Java Development Kit) di Oracle che include anche JRE.*

9 Garanzia di Qualità

Del software non verrà garantita la qualità secondo standard determinati, tuttavia il team intende seguire dei principi di qualità più approssimativi che guidino lo sviluppo del progetto. L'argomento viene approfondito nella sezione 3, *Qualità del Software*, della *Specificazione dei requisiti*.

10 Pacchetti di Lavoro

Possiamo dividere le attività da svolgere in:

- Scrittura e aggiornamento della documentazione
- Implementazione della logica dell'applicazione
- Implementazione del data base locale
- Implementazione dell'interfaccia utente
- Testing e Manutenzione

Maggiori dettagli nel documento *Gestione del Progetto*.

11 Risorse e Budget

Date le dimensioni del progetto e il ristretto personale coinvolto, non è prevista alcuna stima riguardante i requisiti hardware per lo sviluppo, e nessuna previsione sui costi (monetari e temporali) dell'intero processo.

12 Cambiamenti

I cambiamenti più rilevanti, se riscontrati, saranno segnati nei report settimanali come già specificato nella sezione 5 *Attività di Gestione*. In tali documenti sarà segnata la problematica riscontrata, la soluzione sviluppata e (in breve) il ragionamento che ha portato ad essa, così da aiutare con la comprensione nelle fasi di manutenzione future.

13 Consegna

Il prodotto finale e tutta la sua documentazione verranno consegnati attraverso la piattaforma di *GitHub*: agli utenti verrà condiviso l'accesso alla repository del progetto, dalla quale sarà possibile scaricare l'applicativo ed eseguirlo in locale sulla propria macchina.