Università degli studi di Salerno

Corso di Laurea in Informatica

Ingegneria del Software

FreshFood

Studenti:

Anna Fulgione	0512105572
Martina Casalnuovo	0512105110

Anno Accademico: 2019/20

Sommario

1. INTRODUZIONE

1.1 Scopo del sistema

Lo scopo del progetto FreshFood è quello di sviluppare un sito e-commerce per la vendita online di frutta e verdura. Il sistema è stato progettato per permettere a tutte le persone, anche quelle meno esperte dello shopping online, di acquistare i prodotti comodamente da casa.

Per rendere FreshFood raggiungibile facilmente si utilizza la pubblicità tramite canali social.

1.2 Obiettivi di progettazione

Gli obiettivi di progettazione rappresentano le qualità desiderate del sistema e forniscono una serie coerente di criteri che devono essere considerati quando si prendono decisioni di progettazione.

Sono stati identificati i seguenti obiettivi di progettazione:

1.2.1 Mantenimento

- **Estendibilità:** il sistema è implementato in modo tale da poter aggiungere facilmente nuove funzionalità dopo il rilascio.
- **Modificabilità:** FreshFood supporta due tipi di utenti "utente registrato" e "gestore del catalogo". Ognuno di essi ha a disposizione diverse funzionalità.

1.2.2 Utente finale

- **Usabilità:**
  - **Semplice da usare:** le funzionalità del sistema devono essere intuitive. Un'acquirente dovrebbe saper acquistare i prodotti che gli interessano sin da subito.
  - **Consapevolezza dei prodotti acquistati:** per supportare l'acquirente il più possibile durante il suo shopping online, tutte le informazioni relative ai prodotti devono essere visibili all'utente. Si include il prezzo, la descrizione, e la quantità dei prodotti.

1.2.3 Affidabilità

- **Affidabilità:**
  - **Sicura:** il sistema FreshFood dovrebbe essere protetta da crash nel 90% del suo tempo di esecuzione
- **Robustezza:**
  - **Robusto:** Tutte le interazioni tra sistema ed utente devono essere progettate in modo che l'utente non sia in grado di inserire dati non validi.
- **Sicurezza:**
  - **Dati protetti: i** dati degli utenti registrati saranno protetti da chiavi crittografiche, per garantire la massima sicurezza.

1.2.4 Performance

- **Memoria:** i dati del sistema devono occupare 1 GB di memoria sul server.
- **Tempo di risposta:**
  - **Breve tempo di reazione:** il tempo di caricamento del sistema deve essere inferiore a 10 secondi. Tutti gli altri tempi di risposta devono essere inferiori a 2 secondi.

1.3 Definizioni, acronimi e abbreviazioni

1.3.1 Definizioni

FreshFood	Nome del sistema che si vuole sviluppare

1.3.2 Acronimi

RAD	Requirements Analysis Document

1.4 Riferimenti

1.4.1 Relazioni con il Documento di Analisi dei Requisiti (RAD)

Tali relazioni riguardano i requisiti funzionali e non funzionali del sistema.

1.5 Overview

Le sezioni successive di questo documento presenteranno l'architettura del sistema software FreshFood e la sua decomposizione in sottosistemi. Inoltre, verranno indicate le tipologie di utenti, i comportamenti e le funzionalità che il sistema offre ad ognuna di esse. Saranno descritti i requisiti tecnici minimi per l'ambiente e le macchine che ospiteranno il sistema, informazioni sulla gestione dei dati persistenti e le politiche di sicurezza adottate dal sistema.

2. ARCHITETTURA ATTUALE DEL SOFTWARE

Il sistema FreshFood non sostituisce nessun altro sistema già esistente. Per questo motivo, non sono previste modifiche ad alcun altro sistema già sviluppato in precedenza.

3. ARCHITETTURA SOFTWARE PROPOSTA

3.1 Overview

Il sistema FreshFood verrà decomposto in vari sottosistemi, al fine di facilitare il suo design e la sua gestione.

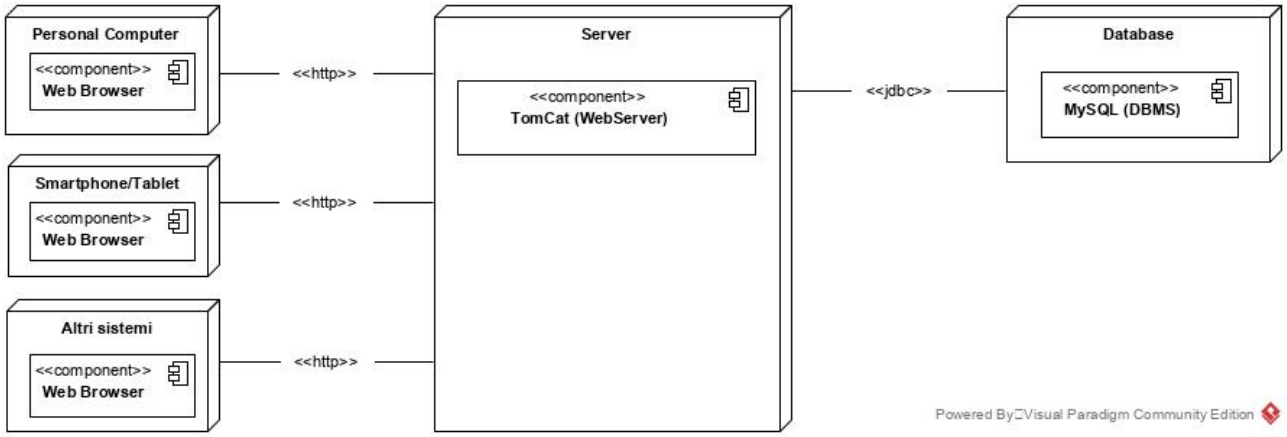
È stata scelta un'architettura Client/Server perché si adatta bene alla scelta dei design goal ed al tipo di software da sviluppare. Infatti, tale architettura favorisce l'affidabilità del sistema, in particolare la robustezza e la sicurezza. Inoltre, essa facilita la progettazione e l'implementazione del sistema per come è stato richiesto dal cliente.

I sistemi Client/Server sono un'evoluzione dei sistemi basati sulla condivisione delle risorse. La presenza di un server permette ad un certo numero di client di condividere tali risorse, lasciando che sia il server a gestire gli accessi ad esse.

Il software client in genere è di limitata complessità, limitandosi normalmente ad operare come interfaccia verso il server. Nel nostro caso, i client saranno rappresentati da tutti i terminali per utenti e per gestori.

Il software server, oltre alla gestione logica del sistema, deve implementare tutte le tecniche di gestione degli accessi, allocazione e rilascio delle risorse, condivisione e sicurezza dei dati o delle risorse.

Il server sarà rappresentato dalla macchina su cui è installato ed utilizzato un DBMS che ne permetterà l'utilizzo per modifiche ed interrogazioni. Inoltre, la macchina server si servirà di opportuni protocolli di trasmissione, come http che permetteranno una corretta comunicazione con i terminali client.



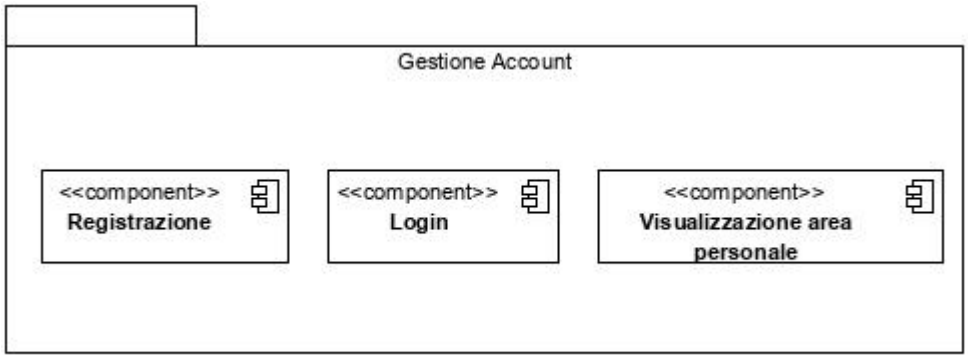
All'interno del sistema verrà utilizzato il design pattern MVC, Model View Controller, metodologia che separa gli oggetti in tre livelli:

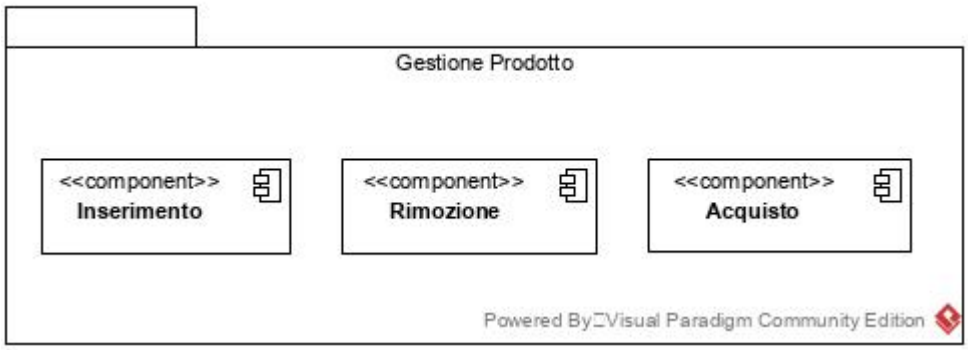
- Model, rappresenta i dati dell'applicazione;
- View, rappresenta la visualizzazione degli oggetti;
- Controller, rappresenta l'insieme di regole che permettono la trasformazione degli input sulle viste in modifiche del modello.

La figura che segue mette in evidenza i vari ruoli svolti dai tre livelli e le interazioni fra essi esistenti. Inoltre la figura esprime esattamente il design pattern applicato alla tecnologia prescelta (Java) per lo sviluppo del sistema software.

3.2 Decomposizione in sottosistemi

La decomposizione in sottosistemi permette di ridurre la complessità del dominio della soluzione. I servizi sono gruppi di funzioni aventi lo stesso obiettivo. L'interazione tra due sottosistemi può essere di vari tipi, ma nel nostro caso è interessante analizzare il tipo Client/Server. Il sistema principale è stato suddiviso in due sottosistemi in modo tale da ridurne la complessità, in base alla metodologia, cercando di tenere alto il livello di coesione nei vari sottosistemi e di tenere basso quello del coupling (accoppiamento) tra sottosistemi distinti. Ciascun sottosistema ha accesso a classi apposite che svolgono le operazioni richieste tramite l'uso di funzioni. Si è cercato di ottenere l'accesso a più funzioni mantenendo comunque una certa dipendenza tra le classi.





3.2.1 Sistema Utente Registrato

Sistema rivolto alle funzionalità dedicate all'utente registrato. Permette di visualizzare ed acquistare i prodotti disponibili nello store direttamente da casa. Inoltre, l'utente ha la possibilità di visualizzare l'area personale e lo storico degli ordini.

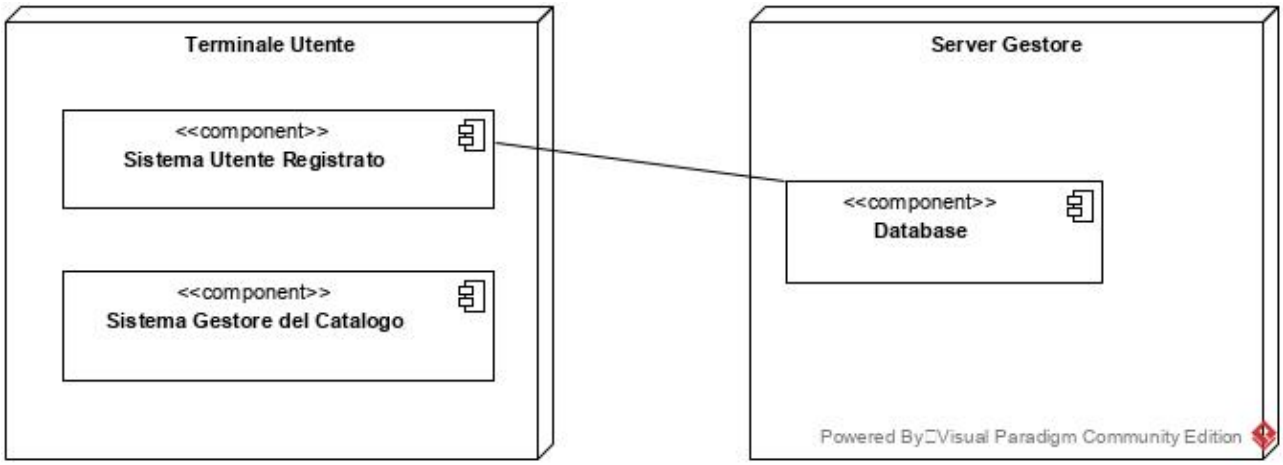
3.2.2 Sistema Prodotto

Sistema indirizzato alle funzionalità dedicate al gestore del catalogo. Permette di visualizzare inserire e cancellare prodotti.

3.3 Mapping Hardware/Software

Il sistema sarà suddiviso in due nodi fondamentali, seguendo i criteri dettati dall'architettura Client/Server:

- Nodo "Terminale Utente": l'utente (utente registrato, gestore del catalogo) accede al sistema attraverso un terminale per usufruire dei servizi offerti dal software
- Nodo "Server Gestore": il database ed il DBMS utili all'archiviazione dei dati



3.3.1 Terminale Utente

L'utente (inteso genericamente) accede al sistema attraverso i terminali, che rappresentano il primo nodo. Sul client risiederà la parte principale del sistema ed esso, in base alle scelte dell'utente ed a login e password inserite, avranno a disposizione funzionalità differenti, a seconda che si tratti di un utente registrato o di un gestore del catalogo.

3.3.2 Server Gestore

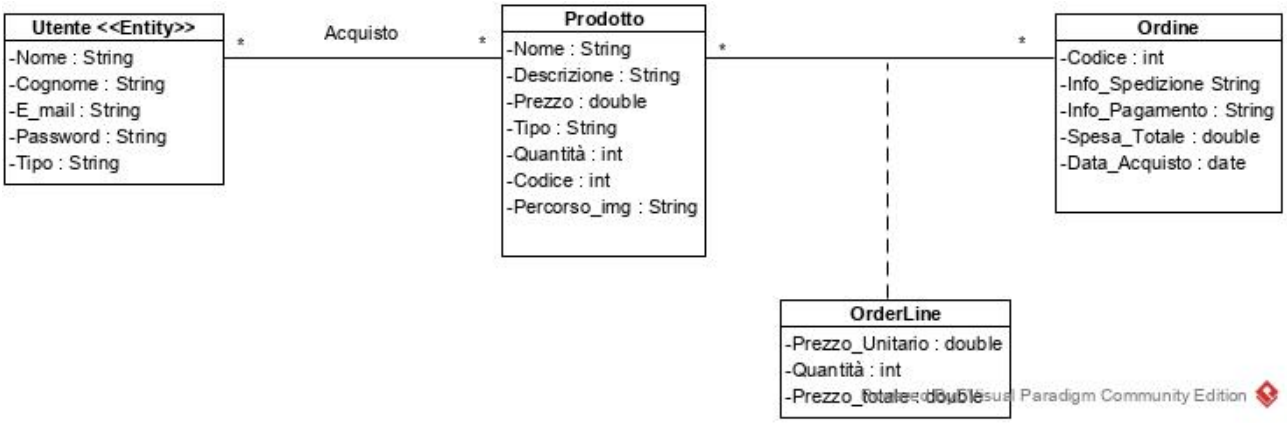
Il secondo nodo è quello dove risiederà il database fisico, contenete tutte le informazioni relative all'intero dominio di applicazione. Su tale macchina devono essere installati alcuni software necessari ai servizi offerti dal sistema, come il DBMS MySql, indispensabile per la gestione dei dati persistenti e come un software per la gestione della connessione con i client.

3.4 Gestione dei dati persistenti

Nelle sottosezioni successive sono illustrati vari modelli che descrivono in vari livelli.

La componente del sistema FreshFood che si occupa di gestire i dati persistenti: il database

3.4.1 Class Diagram



3.4.2 Tabelle database

1. **Prodotto**

Variabili	***Tipo***
Code	int
Nome	Varchar(45)
TipoProdotto	Varchar(45)
Descrizione	Varchar(1000)
Prezzo	double
Quantità	int
Img	Varchar(150)
Stagionalità	Varchar(45)

2. **Utente**

Nome	Varchar(45)
Cognome	Varchar(45)
Email	Varchar(45)
Password	Varchar(45)
Tipo	Varchar(45)

3. **Ordine**

Codice	Int
Utente	Varchar(45)
SpesaTotale	Double
dataOrdine	varchar(45)
datiSpedizione	Varchar(1000)
datiPagamento	Varchar(1000)

4. **Composizione**

Prodotto	int
Quantità	int
Utente	Varchar(45)

I dati persistenti del sito sono gestiti tramite DBMS.

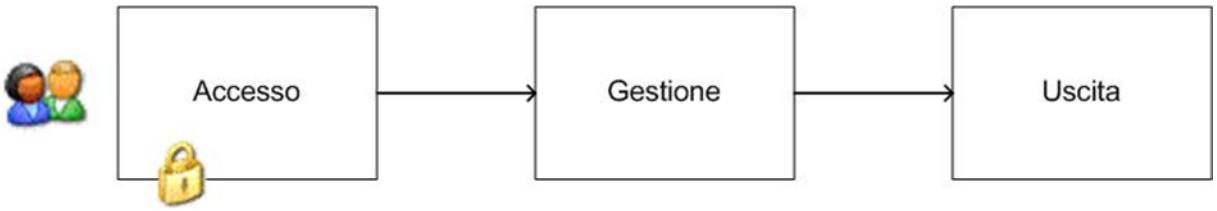
3.5 Controllo degli accessi e sicurezza

Il sistema presenta dispositivi di sicurezza e controlli sugli accessi. Esso prevede l'accesso di tre tipologie di utenti: gestore dei prodotti, utente non registrato, utente registrato. Al sistema è possibile accedere tramite un meccanismo di autenticazione che tramite un form, che richiede l'inserimento di un'e-mail e di una password fornite al momento della registrazione, regola l'accesso al sistema. Quanto detto prima, però, è escluso per l'utente non registrato, che può solo consultare il catalogo dei prodotti. Il sistema, dopo aver analizzato i campi login e password, controllerà che siano presenti nel database. Nel caso in cui l'accesso venga negato, all'utente verrà ripresentato il form per l'inserimento dei dati necessari per autenticarsi. Altrimenti, l'utente potrà avere libero accesso alla propria area.

Funzione	*Utente registrato*	*Utente non registrato*	*Gestore del catalogo*
Gestione dei prodotti	○ Visualizza catalogo ○ Ricerca prodotto ○ Acquista prodotti ○ Visualizza i suoi ordini	○ Visualizza catalogo ○ Ricerca prodotto	○ Inserisce prodotti nel catalogo ○ Rimuove prodotti dal catalogo
Gestione profilo	○ Login ○ Log-out ○ Visualizzazione profilo	○ Registrazione	○ Login ○ Log-out ○ Visualizzazione profilo

3.6 Controllo globale del software

Il flusso di controllo globale del sistema deve essere coordinato all'interno del sistema stesso, lato Client. Il flusso delle operazioni per ogni tipologia di utente è del tipo: Accesso al sistema, Gestione del sistema, Uscita dal sistema, affinché si arrivi ad una esatta e completa esecuzione di un'operazione.



3.7 Boundary Condition

Boundary Condition descrive il comportamento di avvio, arresto del sistema.

3.6.1 Avvio del server

Avvio del server		
ID	BC1	
Attori	Amministratore	
Pre-condizione	L'amministratore accede al server tramite la voce "Avvia"	
Flusso di eventi	**Utente**	**Sistema**
		1-Il sistema si avvia e attiva i servizi in remoto andando a renderli disponibili per le richieste.
Eccezione	Qualora durante la fase di avvio si verificasse un errore, l'amministratore sarà avvisato tramite una notifica	
Post-condizione	L'amministratore accede alla piattaforma	

3.6.2 Arresto del server

Arresto del server		
ID	BC2	
Attori	Amministratore	
Pre-condizione	L'amministratore spegne il server tramite la voce "Stop"	
Flusso di eventi	**Utente**	**Sistema**
		1-Il server controlla che non ci siano richieste in attesa; solo quando non ci saranno più richieste in sospeso il server procede all'arresto
	Qualora durante la fase di arresto si verificasse un errore, l'amministratore sarà avvisato tramite una notifica	
Post-condizione	Il server è spento e i relativi servizi non sono più disponibili.	

4. SERVIZI DEL SOTTOSISTEMA

4.1 Gestione prodotti

Sottosistema che gestisce tutte le operazioni relative ai prodotti disponibili sulla piattaforma. Si distinguono diverse operazioni:

- Inserimento prodotto
- Rimozione prodotto
- Acquisto prodotto

4.2 Gestione degli account

Sottosistema che gestisce tutte le operazioni relative agli account. Si distinguono diverse operazioni:

- Login
- Log-out
- Visualizza profilo