Università degli Studi di Padova
LAUREA IN INFORMATICA

PROGETTO DI PROGRAMMAZIONE AD OGGETTI

"Candival - Sweety Moments"

Studente:

Valentina Signor
matricola: ***1049106***

Anno 2018 - 2019

Sommario

1. SCOPO DEL PROGETTO

Il progetto ha come scopo la creazione di un applicativo finalizzato all'aggiornamento e gestione dei prodotti (chiamati Pasticcerie) disponibili all'interno di un negozio di dolci, e delle sue conseguenti ordinazioni. La visione adottata prevede che sia ad utilizzo del/i gestore/i del negozio e dei corrispettivi clienti.

2. FUNZIONALITA' IMPLEMENTATE

Nello specifico il negoziante (admin) avrà a disposizione un'area riservata a cui avrà accesso dopo essersi autenticato, e da cui potrà:

- creare nuovi sconti;
- visualizzare le varie tipologie di Pasticcerie presenti;
- visualizzare le giacenze in magazzino con possibilità di modifica;
- aggiungere un nuovo sottotipo di Pasticceria: cosa che fa selezionando uno dei tipi disponibili, e inserendo il nome, gli ingredienti ecc.;
- gestione delle ordinazioni (visualizzazione e archiviazione);
- avvio delle finestre clienti (da cui il cliente può ordinare);
- modifica delle credenziali di accesso al sistema;
- aggiornare le informazioni riguardanti il negozio (nome, orari apertura, telefono …);
- possibilità di caricare immagini (che risultano cliccabili), sia a livello di foto profilo che per le varie Pasticcerie presenti.

Il cliente, a cui viene associato un identificativo numerico, potrà invece:

- visualizzare le varie Pasticcerie disponibili;
- fare un ordine e pagare in cassa.

3. PROGETTAZIONE

3.1 Analisi gerarchia principale

Premessa: al fine di rappresentare il più fedelmente possibile una situazione reale si sono individuate quattro grandi classi di dolci (virtuali e non), figlie dirette della classe base polimorfa *Pasticceria*:

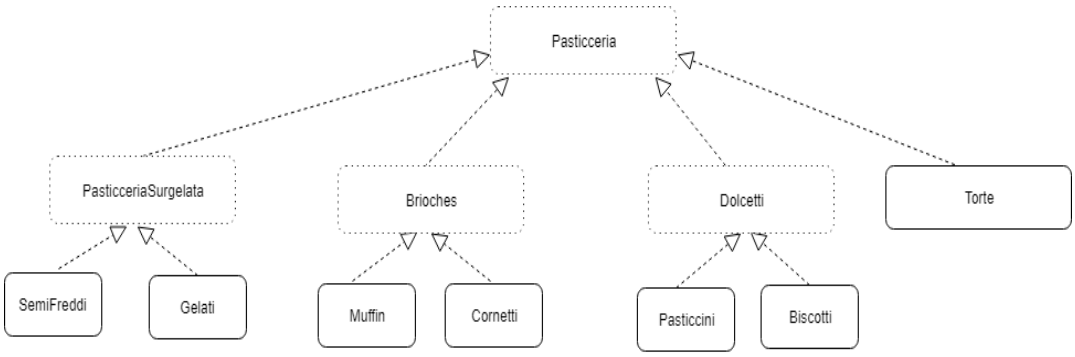

Figura 1: rappresentazione della gerarchia di Pasticceria

PasticceriaSurgelata(prodotti da freezer - virtuale), ***Brioches***(virtuale), ***Dolcetti***(virtuale) e ***Torte***(concreta). Esse genericamente differiscono tra di loro per la composizione (prodotti particolari utilizzati, ripieno, impasto ecc.), la forma e le modalità di mantenimento.

Avendo poi una grande varietà di prodotti, le prime tre a loro volta, danno vita a due classi concrete ciascuna.

Per la categoria *Torte* si è optato invece fosse sufficiente utilizzare un tipo enum (classe Caratteristiche), per distinguerne le varie tipologie.

Scendiamo ora maggiormente nel dettaglio:

In una prima versione della progettazione la classe Pasticceria rappresentava la classe base astratta contenente i metodi virtuali puri necessari alla gerarchia. In una successiva revisione si è deciso di far derivare questa classe dalla classe ***DatabaseData*** che ora rappresenta la vera e propria classe base astratta, genitore delle classi *Pasticceria, Ordini, Sconto ed User*. Essa contiene i metodi virtuali puri e gli identificativi per la comunicazione col database.

Successivamente si presentano le classi:

Pasticceria

Classe base polimorfa in cui si identifica l'insieme di tutti i prodotti presenti nella pasticceria stessa. Essa è caratterizzata da:

una serie di **attributi**, allo scopo di dare alcune informazioni generiche, quali il nome del prodotto, il prezzo, la data di produzione e quella di scadenza, la presenza o meno di qualche caratteristica particolare e così via.

Si evidenzia la presenza di un fields references di tipo intero per il references counting;

un **costruttore** in versione standard;

alcuni **metodi** fra cui quelli concreti di get()/set() necessari per l'inserimento/modifica delle informazioni riguardanti un nuovo prodotto, e i metodi *virtuali* per la clonazione e per la comunicazione col database come completeFromJson e completeInJson

PasticceriaSurgelata – primo passo di derivazione

Classe virtuale che rappresenta l'insieme dei prodotti che devono essere conservati in freezer. *Figli: SemiFreddi e Gelati.*

Essa presenta una serie di **attributi** caratterizzanti quali: gradiMantenimento, pannaFresca e fruttaFresca.

Brioches – primo passo di derivazione

Classe virtuale che rappresenta l'insieme di tutte le possibili brioches che si possono trovare nella pasticceria. *Figli: Muffin e Cornetti.*

Essa presenta una serie di **attributi** caratterizzanti quali: base e ripieno.

Dolcetti – primo passo di derivazione

Classe virtuale che rappresenta l'insieme di tutti i possibili dolcetti che si possono trovare nella pasticceria. *Figli: Pasticcini e Biscotti.*

Essa è caratterizzata da una serie di **attributi** che la specializzano, quali: ripieno, pasta e forma.

Passiamo ora alle *classi concrete*:

Torte

Classe concreta che rappresenta l'insieme di tutte le possibili torte (ad esclusione di quelle che richiedono la conservazione in freezer), che si possono trovare nella pasticceria.

Essa è caratterizzata da una serie di attributi che la specializzano, quali: l (tipologia di appartenenza della torta), s (scopo per cui la torta viene creata), candeline (con o senza).

Semifreddi

Classe concreta che rappresenta l'insieme di tutti i possibili semifreddi che si possono trovare nella pasticceria. Essa è caratterizzata da un singolo **attributo**: fruttaSecca (bool).

Gelati

Classe concreta che rappresenta l'insieme di tutti i possibili gelati che si possono trovare nella pasticceria. Essa è caratterizzata da un singolo **attributo**: forma (in vaschetta, cornetto, ecc.).

Muffin

Classe concreta che rappresenta l'insieme di tutti i possibili muffin che si possono trovare nella pasticceria. Essa è caratterizzata da un singolo **attributo**: yoghurt (bool).

Cornetti

Classe concreta che rappresenta l'insieme di tutti i possibili cornetti che si possono trovare nella pasticceria. Essa è caratterizzata da una coppia di **attributi**: grandezza e forma.

Pasticcini

Classe concreta che rappresenta l'insieme di tutti i possibili pasticcini che si possono trovare nella pasticceria. Essa è caratterizzata da un singolo **attributo**: gelatina(bool).

Biscotti

Classe concreta che rappresenta l'insieme di tutti i possibili biscotti che si possono trovare nella pasticceria. Essa è caratterizzata da un singolo **attributo**: categoria (linea di appartenenza).

Inoltre:

- alcune classi presentano un attributo statico old. Quest'ultimo non è altro che uno sconto comune, ossia vale per tutti gli elementi della classe in cui è previsto, e idealmente è da applicarsi a tutte quelle pasticcerie che risultano essere di vecchia produzione.

Tutte queste classi intermedie o concrete possiedono:

- un costruttore impostato a valori di default;
- dei metodi concreti quali get() e set() e l'override del metodo di clonazione
- metodi necessari per la comunicazione con il database e l'interfaccia (getDetUI, completeFromJson e completeInJson)
- moreDetString per quegli attributi dove è previsto in sede di interfaccia di dare all'utente la possibilità di selezionare tra più voci.

3.2 Classe contenitrice e Deep Pointer

Al fine di gestire al meglio la collezione di prodotti offerti dalla pasticceria è stata appositamente creata una classe contenitrice templetizzata*: **Container***, implementata sotto forma di lista. Lo scopo principale è quello di far fronte alle mansioni di insert, delate, search e di implementare una gestione della memoria senza condivisione (deep memory). Ecco che si evidenziano:

- un puntatore al primo e all'ultimo nodo di questa lista: classe annidata nodo –> first e last;
- vari metodi allo scopo di facilitare l'accesso e l'inserimento in coda e in testa alla lista: begin(), end(), push back...;
- costruttore che inizializza una lista vuota;
- ridefinizione di costruttore di copia, assegnazione e distruzione in modalità profonda;
- presenza sia di un iteratore costante che non, amici della classe contenitrice (con accesso quindi alla sua parte privata);
- metodi come size() che mi fornisce la lunghezza della lista o get() che mi restituisce la lista stessa;
- ridefinizione di alcuni operatori: operator[], operator+=, operator++, operator--, operator*, operator→, operator!=, operator==.

In aiuto alla gerarchia, inoltre si fa uso del ***Deep Pointer***. Esso viene implementato in maniera indipendente (deepptr.h/cpp), con lo scopo di semplificare le operazioni di eliminazione, di assegnazione e di clonazione degli oggetti della gerarchia inseriti all'interno del container, facendo uso del references counting.

3.3 Classe Negozio

Classe concreta che mi rappresenta il negozio in quanto entità fisica e non come insieme di prodotti. Tra le altre cose presenta:

- informazioni generiche sul negozio (nome, il telefono, l'email, l'indirizzo, gli orari ecc.);
- un contenitore di tutti i prodotti presenti nella pasticceria (menu e menuPrincipale);
- un contenitore dei clienti con le relative ordinazioni.

Abbiamo inoltre: un **costruttore** in versione standard e vari metodi **concreti** ciascuno con una sua finalità: manipolare le informazioni del negozio; aggiornare il menù, le ordinazioni e la lista dei clienti; convertire le informazioni per successivamente caricarle nel database; comunicare con le varie view dell'interfaccia.

3.4 Le classi Ordini e Sconto

Tra le attività importanti che l'utente può fare vi sono la gestione delle ordinazioni e degli sconti. Sono quindi state definite appositamente:

- *Classe Ordine:* gestisce ogni singolo ordine commissionato alla pasticceria, con l'indicazione della quantità del prodotto, del cliente commissionario, degli eventuali sconti globali previsti ecc.;
- *Classe Sconto:* cerca di gestire gli sconti totali da applicare sulle ordinazioni, distinguendo fra sconto unitario e sconto su minima quantità.

3.5 Altre classi di supporto

- *Classe Clientref:* riguarda tutte le informazioni e i metodi relativi ad ogni cliente e alle sue ordinazioni;
- *Classe Caratterische*: mi racchiude tutti i tipi enum utilizzati nella gerarchia di Pasticceria;
- *Classe Database*: permette la creazione del database;
- *Classe User*: mi identifica l'utente admin dell'applicativo (necessaria per la creazione del Area Login nell'interfaccia).

4. IL DATABASE

Per la costruzione del database si è fatto uso di Json mediante cui esso si autopopola nel momento in cui vengono fatti i vari inserimenti. Nella Home comparirà così una cartella *ProgettoP2V* che presenterà una sottocartella: *Database* i cui vari files contenuti si aggiorneranno ad ogni intervento dell'utente. In *main.cpp* è inoltre presente un popolamento iniziale.

5. INTERFACCIA UTENTE

Per quanto riguarda l'implementazione della GUI si è cercato di utilizzare in modo consapevole il designer offerto da Qt Creator, implementandolo in parallelo alla normale scrittura tramite codice.

Anzitutto ci si imbatte nella Pagina di Login, a seguito del quale una volta inserite le credenziali, si può accede all'applicativo vero e proprio.

In questa prima finestra si trova il P*annello di Controllo* che dà la possibilità all'amministratore di prendere quattro vie differenti:

- *Pannello Ordini*

- *Pannello Pasticcere*
- *Pannello Negozio*
- *Pannello Utente*

In tutte le pagine interne si presentano: in alto a sinistra il back-button che se premuto fa tornare alla pagina precedente; e in alto a destra un pulsante che, dopo specifica conferma da parte dell'utente, permette direttamente di sloggarsi.

5.1 Pannello Ordini

Una volta cliccato si viene portati in una pagina da cui l'admin può decidere, grazie alla presenza di due appositi pulsanti se:

- Avviare le Finestre Clienti;
- Visualizzare alcune informazioni sull'applicativo

Ha inoltre la possibilità di vedere il numero di ordini in attesa di essere soddisfatti (ordini in coda).

5.2 Pannello Pasticcerie

In tale finestra sono presenti:

- *nella <u>zona centrale</u>*, una serie di immagini cliccabili rappresentative delle quattro famiglie di Pasticcerie offerte dal Negozio. Se cliccate ognuna porta poi alle corrispettive sottofamiglie o direttamente alla categoria di Pasticceria selezionata (nel caso di Torte), per poi giungere alla relativa Scheda Prodotto;

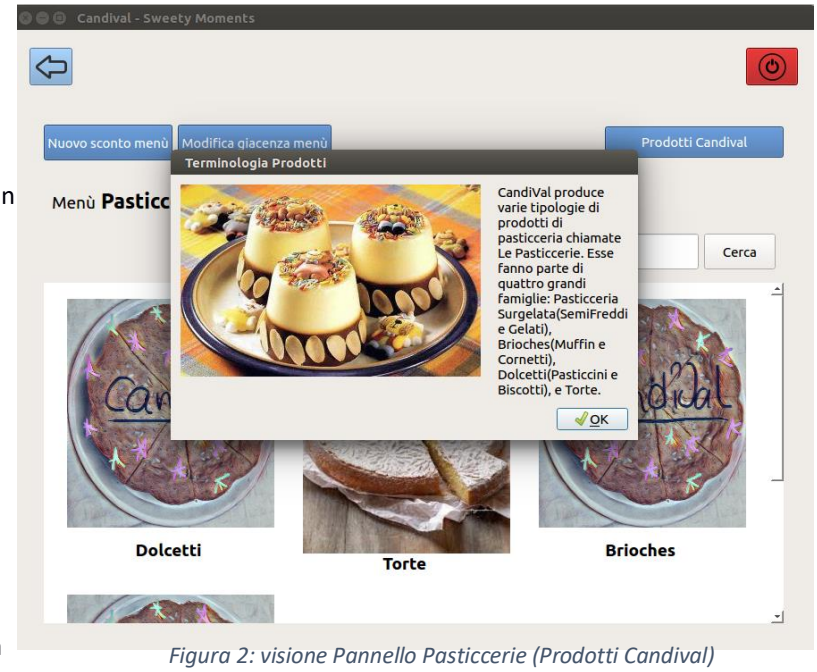

Figura 2: visione Pannello Pasticcerie (Prodotti Candival)

- *<u>in altro</u>*, tre pulsanti attraverso cui è possibile modificare le giacenze in magazzino, introdurre un nuovo sconto e visualizzare alcuni dettagli tecnici sulle tipologie di Pasticcerie offerte. Quando ci si trova ancora nelle categorie intermedie tali implementazioni fanno riferimento in modo generico alla classe intera, se però si clicca in una specifica categoria di prodotti, tali funzioni vengono adattate per quella specifica classe, con in aggiunta le possibilità di andare ad introdurre un nuovo prodotto (nome, ingredienti, prezzo ecc.), e di ritornare al menù principale.

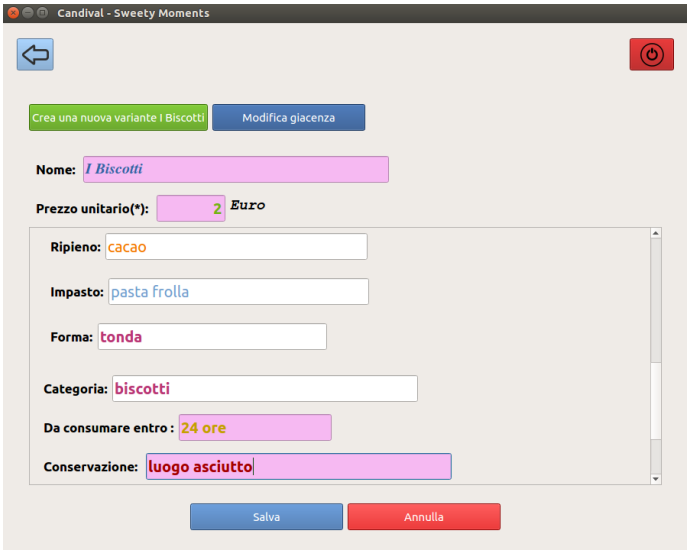

Figura 3: Scheda inserimento Prodotto

Quando viene aggiunta all'interno di una classe una nuova variante del prodotto non indicandovi il nome, essa viene comunque salvata ed etichettata col prezzo.

Per distinguere la classe rappresentativa della categoria da quella che porta alla Scheda Prodotto, il nome di quest'ultima viene preceduto dalla parola *"Classici"* per differenziarla dalla prima.

Per la buona riuscita di un inserimento il prezzo rappresenta un campo indispensabile, inoltre solamente le classi finali danno la possibilità all'utente di caricare un'immagine che le descriva. Quelle intermedie infatti rimangono con il logo stabilito di default.

Grazie ad una ***search*** possiamo andare a ricercare più facilmente le varie Pasticcerie sia nelle classi intermedie che in quelle finali. Riesce ad operare anche solo immettendo parti del nome ricercato e non facendo distinzioni tra minuscole o maiuscole. Se la ricerca fallisce viene consigliato di riprovare venendo reindirizzati al menù di partenza.

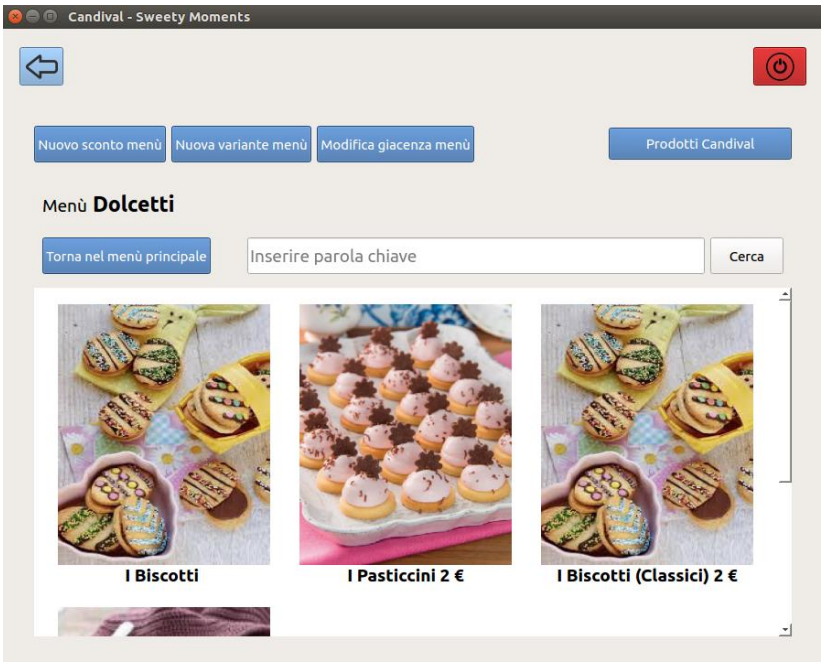

Figura 4: Menù Interno

5.3 Pannello Negozio

Si tratta del profilo del Negozio con tutte le sue relative informazioni. Qui l'utente ha la possibilità di apportare delle modifiche ai vari campi e di selezione e caricare una nuova foto grazie al pulsante "Sfoglia".

5.4 Pannello Utente

Vale lo stesso discorso fatto per il Pannello Negozio solo che riguarda le credenziali utente.

5.5 Finestra Clienti

La Finestra Clienti merita un discorso a parte. E' stato infatti predisposto che il negoziante possa decidere di avviare delle *"Finestre Clienti"* (due per ora) che andranno a comparire in appositi schermi disposti all'interno del negozio. Esse, una volta avviate, presentano il nome, il numero telefonico, l'email e l'indirizzo della Pasticceria, ed è proprio grazie a queste che i clienti dopo aver eseguito l'accesso, potranno visionare le schede dei vari prodotti e inserire le Pasticcerie che preferiscono nel ***carrello***, componendo così la propria ordinazione autonomamente.

Da notare che in ciascuna Scheda Prodotto:

- non tutti i campi sono compilabili dal cliente (è stato settato volontariamente enabled=false);
- presenta l'indicazione degli sconti che vi sono applicati.

Anche qui è stata implementato un box di ricerca per poter selezionare i prodotti con maggiore flessibilità.

E' a discrezione del negoziante decidere quando avviare le Finestre Clienti e quando arrestarle.

5.6 Il Carrello e la gestione degli ordini

E' stata implementata la funzionalità del Carrello dove il cliente può creare il suo ordine con le quantità e le tipologie di Pasticcerie che preferisce. Ogni elemento dell'ordinazione può essere modificato o eliminato in qualunque

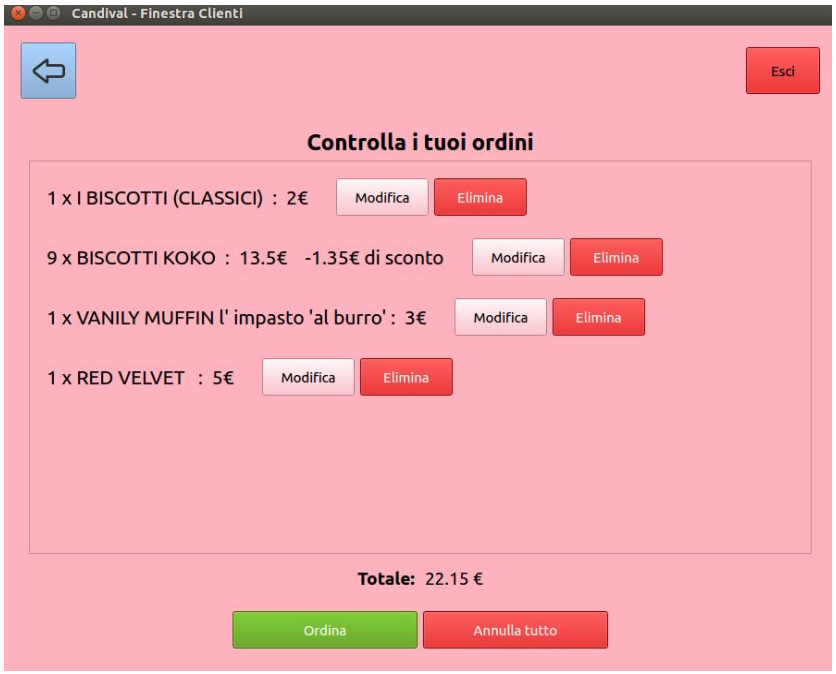

Figura 5: Riepilogo ordinazione (vista cliente)

momento, inoltre una volta visionato il totale, sarà a discrezione del cliente decidere se confermare il tutto o meno. In caso di risposta positiva gli verrà assegnato un numero grazie a cui potrà recarsi in cassa per essere servito e successivamente pagare.

Da notare che passati un paio di secondi dall'assegnazione del numero si è riportati alla Finestra Clienti pronti per la successiva ordinazione.

In parallelo al negoziante compariranno gli ordini in coda, che andrà poi ad etichettare come pagati o non.

5. TEMPO LAVORATIVO E CONCLUSIONI

Le parti che nel complesso hanno richiesto maggiore dispendio temporale sono state quelle relative ad:

- la creazione di un modello che fosse il più flessibile possibile, visto anche il grande numero di tipologie di prodotti e operazioni che in un futuro potrebbero andare ad ampliare la gerarchia. Più volte infatti è stata necessaria una rivisitazione per capire bene cosa andare ad implementare e cosa non (con particolare cura al polimorfismo);
- lo sviluppo dell'interfaccia, per effetto soprattutto del numero di viste che poi si è andato a creare. Si è cercato di fare in modo che la comunicazione fra queste ultime e il modello fosse il più semplice ma efficiente possibile, inoltre come accennato in precedenza si è utilizzato anche Qt Creator per generare alcuni file *ui* che fossero da supporto ad alcune parti del codice e non ad uso esclusivo.

ATTIVITA' SVOLTA	***TEMPO IMPIEGATO***
Analisi del problema	6 ore
Progettazione	4 ore
Codifica e implementazione del modello/ della gerarchia polimorfa	18 ore
Implementazione e codifica GUI (incluso apprendimento libreria Qt)	25 ore
Test generali e operazioni di Debug	5 ore
Stesura relazione	3 ore
ORE TOTALI	**61 ore**

Nel complesso si sono spese un numero di ore leggermente superiore a ciò che inizialmente era stato preventivato, soprattutto a causa della mancanza di conoscenze iniziali della libreria Qt che ha richiesto il suo tempo per essere compresa.

Da evidenziare inoltre la presenza dei metodi per poter gestire sia le ordinazioni (distinte fra pagate, non pagate e servite) che gli sconti complessivi, anche se poi a livello di interfaccia non sono state implementate direttamente.

6. CREDENZIALI DI ACCESSO

Username: vale

password: admin

7. AMBIENTE DI SVILUPPO UTILIZZATO

Per sviluppare l'applicativo si è fatto so di:

- **Sistema Operativo di Sviluppo:** Ubuntu 16.04 LTS;
- **Versione della libreria c++:** Qt creator nella versione 5.9.5;
- **Versione del Compilatore c++:** GCC 4.6.1, 64 bit.

8. COMPILAZIONE ED ESECUZIONE DEI FILE SORGENTE

La gerarchia delle varie directory è la seguente:

progettoP2 è la cartella contenente il progetto. Essa si articola in:

- *ProgettoPasticceria* che a sua volta si compone delle cartelle:
  - ***Modello,*** contenente tutti i files .cpp e .h riguardanti l'implementazione del modello;
  - **View**, contenente tutti i files riguardanti le viste. Qui a sua volta è contenuta una sottocartella, ***ui*** che racchiude tutti i files implementati utilizzando il Designer di Qt Creator;
  - ***images***, contenente tutte le immagini caricate all'interno dell'applicativo;
  - ***main.cpp***;
  - ***ProgettoPasticceria.pro***;
  - ***risorse.qrc***.
- *relazione.pdf* contenente la relazione richiesta.

Da terminale spostarsi all'interno della directory *ProgettoPasticceria* e digitare il comando: **qmake** per produrre il Makefile. Successivamente si può procedere alla compilazione mediante il comando: **make.** E' stato fornito il file *ProgettoPasticceria.pro* in quanto si necessita di includere i moduli di qwidget.
Per eseguire il programma correttamente digitare da terminale: **./ProgettoPasticceria**.