Restaurant Manager

progetto: Gestore di un ristorante

September 11, 2022

Abstract

Un progetto di Coppola Armando, D'Aprile Francesco e Di Santo Niccolò.

1 Introduzione

Il nostro progetto, titolo definitivo, presenta un programma per la gestione di un ristorante. Ogni ruolo all'interno dell'attività ha la propria schermata di gestione. I ruoli per lo svolgimento del progetto sono stati divisi nel seguente modo

Armando Coppola:

- backend
  - creazione piatti e gestione menu
  - creazione ordini e gestione ordini
- frontend
  - Interfaccia grafica dello chef
  - parti dell'interfaccia del cameriere
- relazione

Francesco D'Aprile:

- frontend
  - interfaccia grafica del cuoco e annessi componenti
  - parti dell'interfaccia del cameriere
- UML

Niccolò Di Santo:

- interfaccia mainmenu
- interfaccia grafica della cassa
- classe receipt
- relazione

Le descrizioni delle classi che seguiranno verranno strutturate nel modo seguente:

-breve descrizione della funzionalità della classe partendo dalle classi backend per poi parlare delle classi frontend

-lista degli attributi

-spiegazione dei metodi

-lista dei metodi getters (metodi che servono per ricavare gli attributi dai vari oggetti)

-lista dei metodi setters (metodi utilizzati per modificare gli attributi degli oggetti)

2 librerie utilizzate

Gson-2.8.2
La libreria Gson è stata utilizzata per creare stringe **.json** e per ricreare oggetti da stringe **.json**

3 Descrizione delle classi

3.1 pacchetto backend

Dish Questa classe serve, tramite l'input degli appositi parametri, a creare l'oggetto piatto (Dish) e poterne anche ricavare gli attributi. Attributi: name; category; price; description; avaiable. Con il costruttore di questa classe si ha un caso di polimorfismo. Difatti vi è un overloading con il costruttore che necessita in input i parametri che verranno tramutati in tutti gli attributi, ma se il parametro "avaiable" non fosse inserito in input, allora il secondo costruttore lo imposterebbe a true. I metodi getters, ovvero i metodi che ritornano un dato preciso presente nella classe sono: getName, getPrice, getCategory, getDescription, isAvaiable. I metodi setters, ovvero i metodi che al contrario non ritornano dati ma li impostano o cambiano, invece sono: setName, setAvaible, switchaviable, setDescription, setPrice, setCategory ed in fine il metodo toString che ritorna il nome del piatto in formato stringa.

DishMenu Questa classe serve a creare il menu del ristorante, aggiungerci e toglierci piatti e riscriverlo in json. L'unico attributo presente è "menu", l'Hashmap con una stringa come chiave e l'arraylist con i piatti come valore. Il primo metodo è "add" e prende in input un oggetto Dish (dalla classe Dish), e ne rileva la categoria, che se non presente nella lista del menu, la aggiunge ed in seguito aggiunge il piatto, altrimenti se presente, aggiunge unicamente il piatto nel menu. Dopo il metodo add, abbiamo il metodo "removeDish" che rimuove un piatto dal menu. Se rimuovendo il piatto la categoria di esso rimane vuota, il metodo si preoccupa di cancellare anche la categoria dal menu. Questo metodo prende in input un oggetto piatto e ritorna una variabile booleana a seconda se la rimozione del piatto è riuscita o meno. In seguito vi è il metodo getDish, che prende in input le stringe nome e categoria e ritorna, se esistente, l'oggetto piatto, cercato tramite le due stringe di input. Con questo metodo troviamo un caso di polimorfismo, dato che è possibile, tramite un secondo metodo getDish, cercare un piatto solo scrivendo in input il nome di esso. Dopo questo parallelismo abbiamo il metodo "toArrayList" che crea un arraylist appunto, contenente tutti i piatti del menu. Gli ultimi tre metodi della classe DishMenu sono metodi connessi, ovvero il primo è il metodo "toJson" che crea una stringa formato json con il menu a Hashmap, da salvare; poi vi è il metodo "save" che salva, tramite BufferedWriter, in un file la stringa json e infine vi è il metodo load che tramite BufferedReader, legge il file salvato con "save" e lo trascrive nel menu finale. In fine il costruttore di DishMenu non richiede parametri ma crea un nuovo menu.

Order La classe Order serve per preparare l'oggetto ordine (Order), che verrà poi elaborato dalla classe OrderManager. La creazione dell'ordine avviene con l'assegniazione di un piatto ad un tavolo, con l'aggiunta di eventuali note. Gli attributi di questa classe sono: dishName; dishPrice; state; table; note; states[ ] (array di Stringe contente tre elementi, stati possibili degli ordini: "preparation", "ready", "delivered"). Anche in questo caso con il costruttore abbiamo un caso di overloading, dunque il primo costruttore di Order riceve in input l'oggetto Dish; l'intero table, e una String note. Dopodichè dall'oggetto Dish vengono prelevati (tramite due getters) il nome e il prezzo del piatto che diventano i parametri dell'oggetto Order. L'intero "table" e la stringa "note" diventano direttamente gli attributi dell'oggetto e "state" viene impostato a 0. Il secondo costruttore invece al posto dell'oggetto Dish accetta come parametro: una stringa dishName, un intero dishPrice, e poi sempre table e note. Dunque dishName e dishPrice diventano gli attributi della classe insieme a table e a note. Dopodiché ci sono tre casi di Overriding, il primo attraverso il metodo "toString" per mutare l'ordine in una stringa; il secondo con il metodo "equals" dove si ritorna una variabile booleana ed il terzo caso di overriding avviene con il metodo "hashcode", il quale ritorna un intero che rappresenta la sequenza hash di (dishName, state, table, note). I metodi getters presenti sono uno per ogni attributo ed i metodi setters invece: setState, che setta lo stato dell'ordine e setNextState, che imposta automaticamente il prossimo stato, dopo l'ultimo stato ricomincia da capo.

OrderManager Questa classe, grazie all'utilizzo della classe Order, gestisce gli ordini e attribuisce a ogni tavolo una lista con i propri ordini, così che al momento del pagamento sia tutto più rapido. Attributi: un Hashmap "register" privata che ha come chiave l'intero del tavolo e come valore un Arraylist contenente tutti oggetti Order.

Il costruttore di questa classe non ritorna nulla e subito dopo di esso vi è il metodo add che serve per ad aggiungere un nuovo ordine al tavolo e se erano già associati alcuni ordini al tavolo il nuovo rodine varrà concatenato agli altri.

Dopo il metodo add si ha un ulteriore episodio di overloading dove, con il metodo getTableList che ritorna un'Arraylist di tutti i tavoli, il primo metodo non riceve nessun dato in input, il secondo metodo invece riceve in input la stringa "state", dunque serve a filtrare l'output della lista dei tavoli con la lista dei tavoli aventi quello specifico stato. In seguito si ha il metodo getOrdersToDeliver che ritorna la un Arraylist di Ordini pronti per essere portati al tavolo. Poi abbiamo i metodi toJson, save e load per tenere traccia in un file json (creato nuovo se non presente), dello storico di tutti gli ordini. In fine ci sta il metodo cleanTable per quando un tavolo si libera e deve essere tolto dunque dal registro. Metodi getters: solo un metodo che ritorna registro di ordini. Metodi setters: sempre un metodo che prende in input "register" che viene quindi mutato nell'attributo unico di questa classe.

OrderPreview La classe OrderPreview estende Order, dunque ne eredità tutti i metodi ed attributi, mostra la quantità di un ordine preciso di un tavolo. Attributi: quantity, ovvero la quantità di un ordine.

Il costruttore prende in input l'oggetto Dish, l'intero table e string note. e aggiunge la quantità di quel piatto per il tavolo preso in input. Gli altri metodi sono: setQuantity, per impostare una quantità, increment, per incrementare di uno la quantità, decrement, per decrementare la quantità, getQuantity, per ricavarla, toOrder crea un oggetto Order con i parametri del costruttore di OrderPreview ed in fine vi è un override con il metodo toString per ritornare la quantità di un piatto con eventuali note, come stringa.

PreviewsRegistrer Questa classe serve a mostrare in anteprima le ordinazioni di un tavolo prima di essere mandate in cucina. È una classe con un solo attributo, un arraylist di oggetti "Orderpreview", "previews". I metodi presenti sono per esempio il metodo "increment" il quale, incrementa la quantità di ordinazioni specifiche quando mostrate in anteprima, oppure parallelamente opposto vi è il metodo "decrement" ma in caso la quantità di quello specifico ordine diventi nulla, ne toglie l'anteprima. Poi vi sono i metodi "addOrder", "clear", e "toOrders", i quali, addOrder aggiunge l'anteprima di un ordine o se già presente ne aumenta la quantità, clear svuota tutte le ateprime presenti e toOrders ritorna un arraylist con tutti gli ordini effettuati e le loro quantità. L'unica classe getter è la classe "getPreviews" che ritorna appunto la variabile previews.

Receipt Ultima classe backend con 13 attributi, per stampare il preconto ed in fine lo scontrino una volta aperto il menu della cassa. I 13 attributi sono: un intero numero del tavolo; un double totale del conto, double importo; stringa testo stampato sullo scontrino; Arraylist con gli ordini dalla classe "Orders"; una double finale con l'Iva; una stringa con il titolo del ristorante; un intero con la lunghezza della stringa; una stringa title con i dati del mercante; una stringa con la directory nella quale salvare gli scontrini; una stringa nella quale il titolo del conto viene letto e scritto e due stringe con la data e l'ora del momento in cui viene stampato lo scontrino. Il costruttore di questa classe è presente ed inizializza praticamente tutte le variabili. I primi metodi che possiamo trovare sono "setTitle" che scrive il nuovo titolo nel file e lo ricarica, poi vi è il metodo "loadTitle" che carica il titolo dal file e se non esiste, ne imposta uno predefinito; un altro metodo è il "addOrders" che da la possibilità di aggiungere ordini direttamente sullo scontrino; poi possiamo trovare il metodo "writeRecipt" che scrive il preconto del tavolo che sta per pagare ed in seguito il metodo "writeEndReciept" stampa lo scontrino vero e proprio una volta avvenuto il pagamento. L'ultimo metodo decisivo è il metodo save che salva lo scontrino nella directory impostata dal costruttore. Vi sono ancora quattro metodi che ritornano attributi o variabili che sono: i metodi getters "getTotal" e "getRecieptText" per far ritornare il totale del conto e il testo dello scontrino ed in fine i metodi "enterAmount" che imposta l'attributo importo(amount) al numero preso in input e "giveChange" che ritorna l'eventuale resto dell'importo.

3.2 pacchetto Frontend

Le classi frontend presenti si suddividono in componenti e finestre.

3.2.1 pacchetto Component

BackMainMenuButton estende JButton allo scopo di chiudere la finestra in cui si trova e aprire la schermata del menù principale il costruttore accetta un parametro di tipo JFrame che rappresenta il frame nel quale viene inserito il bottone per farlo è stato aggiunto un ActionListener che crea un nuovo frame di tipo MainMenu e chiude il JFrame passato come parametro al costruttore della classe

BackgroundPanel estende JPanel allo scopo di inserire un'immagine nello sfondo del pannello con l'overload del costruttore possiamo avere un costruttore che non ha parametri e carica un'immagine impostata di default ed un costruttore che accetta come parametro il path di un'immagine da caricare.
il metodo privato void **loadImage()** prende come parametro l'immagine e la carica in memoria bloccando l'esecuzione del codice fino a caricamento completo.
il metodo protetto void **paintComponent()** è un override del metodo di JPanel che serve ad impostare lo sfondo al pannello

CheckBokPopp Questa classe estende JFrame, viene visualizzata quando viene selezionato un tavolo all'interno dell'interfaccia grafica del cuoco. Questa finestra contiene tutti i piatti da preparare del tavolo con le relative note. I piatti preparati possono essere spuntati per far risultare al cameriere che sono pronti per la consegna. Alla chiusura si riapre la finestra del cuoco aggiornando la situazione dei tavoli; per esempio se tutti i piatti di un tavolo sono stati preparati quel tavolo non apparirà al cuoco.

NumberField estende JField per accettare solo caratteri numerici. nel costruttore viene inizializzato un key listener che legge ogni carattere digitato e lo cancella se non si tratta di un numero o di un punto. possiede poi un metodo "getInt" e un metodo "getDouble" che ritornano il valore all'interno del field convertito in int o in double. possiede un metodo "isDouble" che provando a convertire il testo nel field ritorna un valore booleano che ca se si tratta di un numero oppure no.

TableButton Questa classe estende JButton, è un bottone con lo stile predefinito che indicherà ogni tavolo all'interno dell'interfaccia del cuoco. Quando un TableButton viene premuto, si apre un CheckBoxPopup con le informazioni del tavolo che è stato selezionato.

TabelPanelColored estende un jpanel per colorarlo in funzione degli ordini contenuti al suo interno.

- Verde - tutti gli ordini sono da preparare
- Giallo - alcuni ordini del tavolo sono pronti
- Rosso - tutti gli ordini del tavolo sono pronti

3.2.2 pacchetto windows

CashFrame Questa è la classe dell'interfaccia grafica del menu Cash, che tratta i conti e stampa gli scontrini dei tavoli. Estende StandardFrame. Il costruttore crea un nuovo oggetto CashFrame. Il metodo **init** inizializza i componenti del frame tra questi, vari panel per i testi, il bottone per stampare lo scontrino e quello per cambiare il titolo. Poi vi è il metodo **showReceipt** che mostra nell'area di testo lo scontrino corrispondente al tavolo selezionato. Dopo showReceipt c'è **printReceipt** che stampa lo scontrino finale su schermo e dopo aver controllato se l'importo è inserito e sufficiente, lo salva in fomrato '.txt'. L'ultimo metodo **setTitle** serve per creare il titolo modificato dello scontrino.

ChefFrame La classe ChefFrame è l'interfaccia grafica del menu Chef, che gestisce il menu del ristorante estende StandardFrame. In questa classe il costruttore si limita a creare il frame Chef. Poi vi è il metodo **init** che inizializza i vari componenti come panel e bottoni. Poi dopo abbiamo il metodo **resetInput** che imposta tutti i campi e le aree di testo con stringhe descrittive o utili. Poi invece **setTextAreas** imposta le aree di testo con le informazioni del piatto selezionato. D'altra parte vi è **manageAddButton** per gestire il comportamento del pulsante e fornisce consigli su come aggiungere un nuovo piatto. Un metodo molto importante è **saveProtocol** che gestisce tutti i salvataggi di Chef e riconosce se un piatto è stato aggiunto, eliminato o semplicemente modificato. L'ultimo attributo **removeProtocol**si preoccupa di rimuovere il piatto selezionato dal menù del ristorante.

CookFrame Classe dell'interfaccia grafica di Cook lavora con gli ordini pronti o ancora da preparare.
questa classe ha un solo attributo "pt", un pannello con tutti i tavoli con almeno un ordine non pronto. Il costruttore crea la finestra Cook e il metodo **initComponents** prende le informazioni dei tavoli e imposta le dimensioni e lo stile. Il metodo **clear** ripulisce la pagina e il metodo **Listener**, che implementa "ActionListener" e "WindowListener" crea le interfacce dei tavoli. Gli ultimi metodi sono degli override che resettano le informazioni dei tavoli una volta chiusa la loro interfaccia.

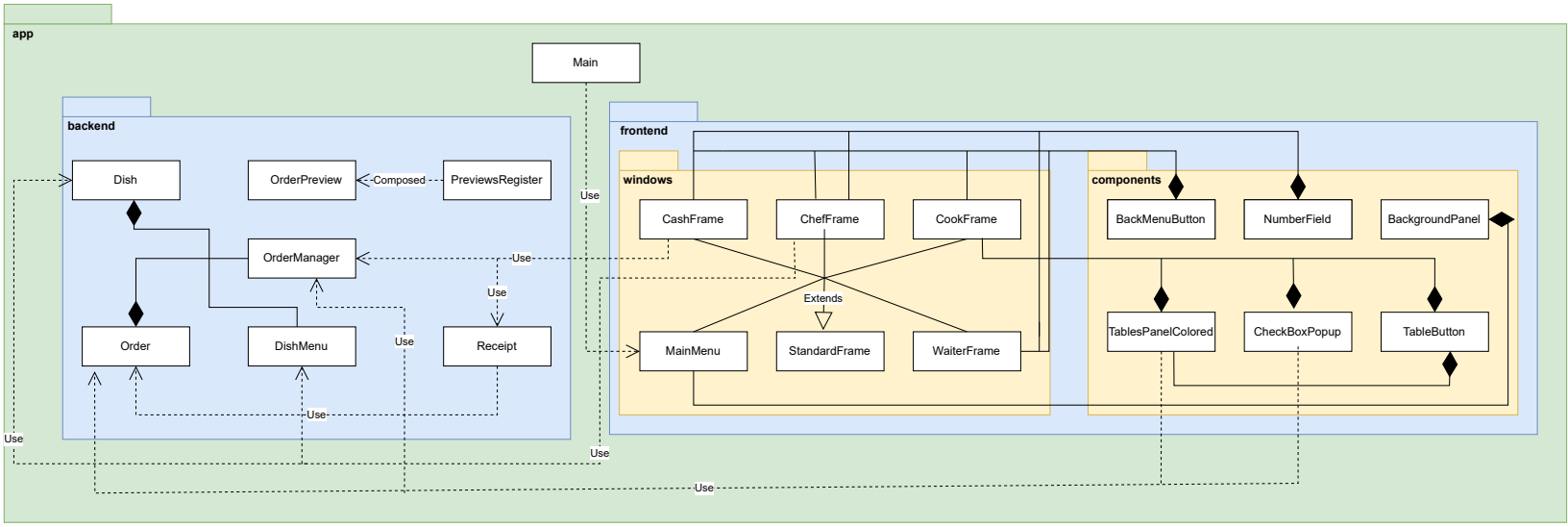
MainMenu È il menu principale del progetto, nel quale è possibile selezionare il ruolo. estende StandardFrame e implementa ActionListener, i vari attributi sono: btnWaiter, il bottone del menu Waiter; btnChef;btnCook;btnCash e ratio un valore predefinito che aiuta con il ridimensionamento delle finestre. Il costruttore crea l'interfaccia, **initComponents** inizializza i componenti e gli ultimi metodi override aprono la schermata selezionata e chiudono questo menu.

StandardFrame Questa classe è un frame standard con attributi in comune ad altre finestre Estende Jframe e ha gli attributi di altezza e larghezza preimpostati a 1080x720 e un attributo "dimension" che li usa. L'unico metodo è il costruttore che crea un nuovo StandardFrame definendone lo stile e le dimensioni.

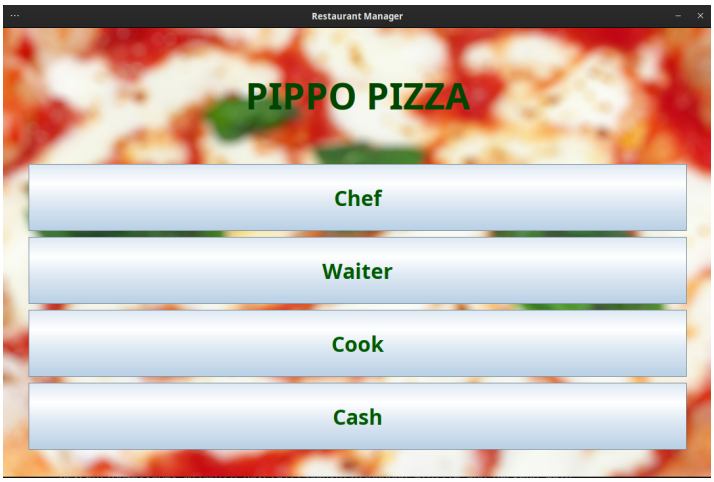
WaiterFrame È l'interfaccia grafica di Waiter, che spedisce gli ordini da preparare e rimuove quelli consegnati WaiterFrame estende StandardFrame e ha come attributi: previewsRegister, il registro temporaneo dove vanno gli ordini non ancora inviati alla cucina, orderManager, numberField, la casella nella quale inserire il tavolo che sta prendendo l'ordinazione, deliverList, listMenu e listPreview. Il costruttore di Waiter crea la finestra aggiunge le informazioni necessarie e imposta lo stile. **init** inizializza il tutti i componenti, **showDish** invece apre una JOptionPane per trovare le informazioni dei piatti, aggiungere eventuali note e aggiungere il piatto nella lista dei piatti temporanei, non ancora spediti in cucina. Poi c'è **placeOrders** che aggiunge gli ordini al registro temporaneo, **changeOrderQuantity** che aumenta o decrementa la quantità di un piatto ed infine **deliverOrder** cambia lo stato stato dell'ordinazione selezionata segnalando che è stata spedita.

4 UML delle classi

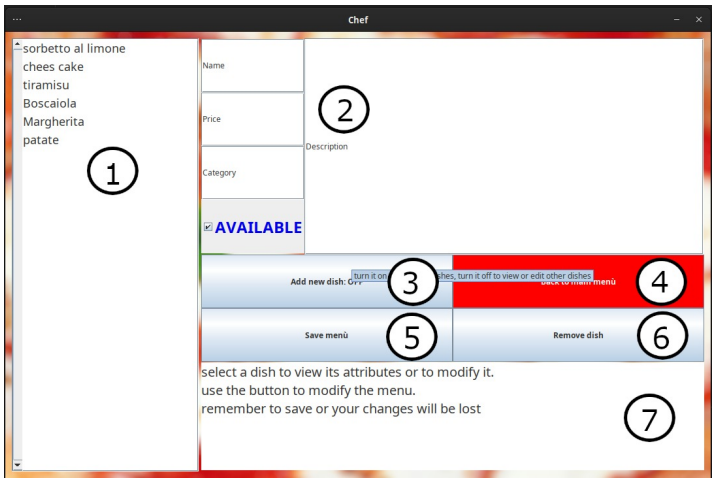
segue la gerarchia UML delle classi


app
Main
backend
Dish
OrderPreview
Composed
PreviewsRegister
OrderManager
Order
DishMenu
Receipt
Use
frontend
windows
CashFrame
ChefFrame
CookFrame
MainMenu
StandardFrame
WaiterFrame
Extends
components
BackMenuButton
NumberField
BackgroundPanel
TablesPanelColored
CheckBoxPopup
TableButton

5 manuale della grafica

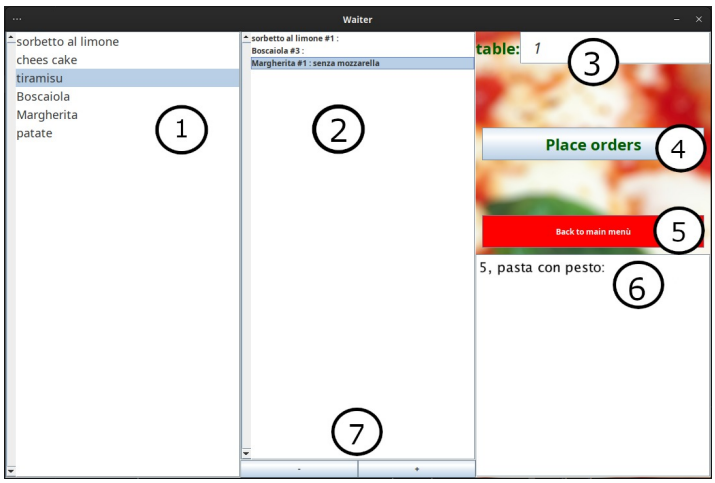


5.1 Menu Chef



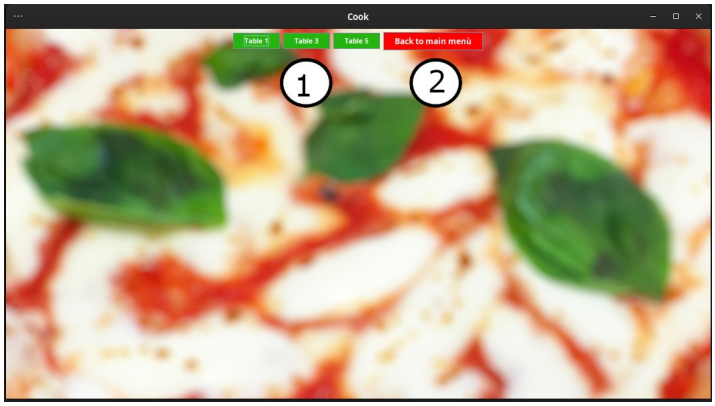
1. : Il menù del ristorante sotto forma di lista, ogni piatto è selezionabile
2. : Caselle di testo modificabili nelle quali si trovano tutte le informazioni necessarie sul piatto selezionato
3. : Bottone per aggiungere un nuovo piatto
4. : Bottone per tornare all'interfaccia principale
5. : Bottone per salvare le modifiche fatte al menù
6. : Bottone per rimuovere un piatto dal menù
7. : Casella di testo nella quale sono scritte brevi istruzioni per l'uso di quest'interfaccia

5.2 Menu Waiter



1. : Il menù del ristorante sotto forma di lista, ogni piatto è selezionabile
2. : Lista temporanea che contiene le ordinazioni ancora in svolgimento e per questo non ancora passate alla cucina
3. : Casella nella quale inserire il numero del tavolo che sta effettuando l'ordinazione
4. : Bottone per passare le ordinazioni direttamente in cucina
5. : Bottone per tornare all'interfaccia principale
6. : Lista con gli ordini pronti, se selezionati passano allo stato "consegnato"
7. : Bottoni più e meno per aumentare o decrementare la quantità di ordini di un piatto

5.3 Menu Cook

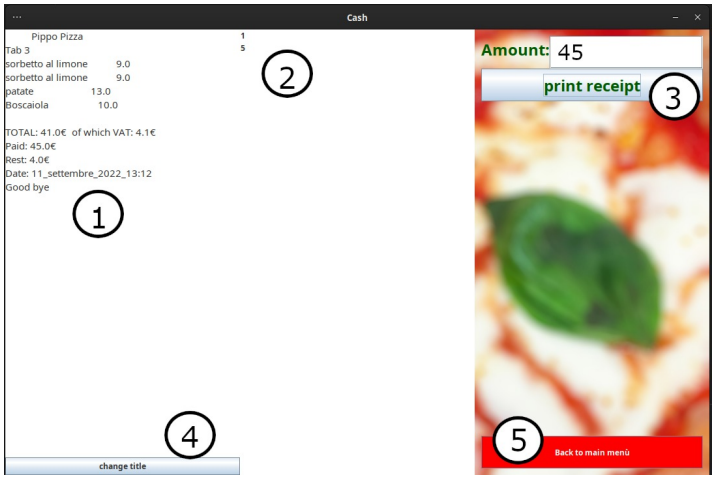


1. : Bottoni che rappresentano ognuno un tavolo con ordinazioni non ancora pronte
2. : Bottone per tornare al menu principale

Tavolo all'interno del menu Cook

1. : Lista degli ordini non ancora pronti
2. : Bottone per vedere le note aggiunte alle ordinazioni
3. : Bottone di conferma, se premuto cambia lo stato di tutte le ordinazioni spuntate a "pronto"

5.4 Menu Cash



1. : Scontrino finale del tavolo selezionato
2. : Lista di tavoli con ordinazioni tutte allo stato "consegnato"
3. : Casella di testo dove inserire il totale dell'importo e bottone per stampare(salvare) lo scontrino
4. : Bottone per cambiare il titolo dello scontrino
5. : Bottone per tornare al menu principale

6 Funzionalità del programma

6.1 gestione menu

il programma gestisce la creazione e la modifica di un menù di piatti creati dall'utente

inserimento piatti tramite l'interfaccia grafica l'utente comunica al programma l'intenzione di voler creare un nuovo piatto, poi inserisce tutte le caratteristiche richieste (nome del piatto, prezzo, categoria, descrizione e disponibilità) negli appositi spazi, infine salva la modifica del menù.

rimozione piatti l'utente seleziona il piatto che vuole rimuovere clicca sull'apposito bottone e il programma rimuove il piatto dall'HashMap. il file però non verrà riscritto fino a quando l'utente non salverà le modifiche così da rendere la rimozione del piatto reversibile.

modifica piatti quando l'utente seleziona un piatto dal menu, i suoi attributi vengono visualizzati nei field dell'interfaccia. se questi vengono modificati quando l'utente salva le modifiche il programma elimina il piatto selezionato e lo rimpiazza con uno nuovo creato con gli attributi presi dai field. Ricreare il piatto all'atto pratico si rivela una scelta più rapida e sicura di modificare singolarmente ogni attributo del piatto selezionato.

salvataggio dati i dati del programma non vengono eliminati quando il programma viene chiuso, ma bensì vengono salvati in file json che sintetizzano gli attributi degli oggetti da salvare. In questo modo il menù, il registro degli ordini e l'intestazione dello scontrino(salvato come .txt) non devono essere ricreati ogni volta che si apre il programma.

scrittura dati tramite l'uso del metodo toJson si crea la stringa json che racchiude gli attributi da salvare e tramite il metodo save viene scritto il file nell'apposita cartella

lettura dati vengono letti i dati dai file, viene convertito il json nell'oggetto necessario e vengono settati gli attributi ai valori caricati.

6.2 gestione ordinazioni

inserimento ordine è possibile inserire ordini per un tavolo creando un'anteprima della comanda modificabile nelle quantità.
Nel ruolo di **Waiter** quando si seleziona un piatto dal menù viene visualizzata una scheda che presenta le informazioni sul piatto e nel quale è possibile inserire eventuali note aggiuntive sull'ordinazione visualizzabili dal cuoco.

gestione quantità Mentre si sta facendo l'ordinazione, è possibile aumentare la quantità dei piatti richiesti semplicemente premendo il tasto "+" sotto le ordinazioni e il previewsRegister verrà aggiornato con la quantità di un determinato piatto richieste. Per diminuirne la quantità invece, basta premere sul tasto "-", se con la diminuzione del piatto si arriva a zero quantità, il piatto viene rimosso dal registro.

inserimento ordini nel sistema una volta conclusa l'ordinazione e premuto il tasto "place orders" le ordinazioni verranno salvate dall'orderManager e diventano visibili per la cucina.
La lista dell'anteprima dell'ordine viene svuotata.

6.3 stato ordini

ogni ordinazione ha un attributo chiamato stato che lo definisce in tre possibili condizioni:

1. in preparazione
2. pronto

3. consegnato

All'interno del programma è possibile:

visualizzazione tavoli con ordinazioni Tutti i tavoli che hanno effettuato delle ordinazioni verranno visualizzati nell'interfaccia del **cuoco** e se un tavolo dovesse aver concluso le ordinazioni da preparare allora verrebbe tolto dalla schermata.

spuntare gli ordini pronti L'utente nel ruolo di **cuoco** appena l'ordinazione è pronta la può spuntare come "pronta" in questo modo il suo stato cambia e ciò viene segnalato ai camerieri che dovranno consegnarla

visualizzazione ordini pronti nel ruolo di **cameriere** è possibile vedere gli ordini pronti da essere consegnati ai tavoli. Selezionandoli questi cambiano di di stato da pronto a consegnato e non vengono più visti dal cameriere.

6.4 scontrino

Nel ruolo di **cassa** è possibile:

visualizzazione ordinazioni dei tavoli visualizzare una lista di tutti i tavoli con ordinazioni. Selezionando un tavolo sarà verrà generata e visualizzata l'anteprima dello scontrino con l'importo totale.

generazione scontrino Inserendo l'importo ricevuto dal cliente è possibile generare lo scontrino completo di importo ricevuto, resto, data e ora della generazione dello scontino.

salvataggio scontrino una volta generato lo scontrino il programma lo salva come testo (formato ".txt") in una cartella chiamata "Receipts" all'interno della quale è possibile accedere a tutti gli scontrini generati dal programma.

cambiare intestazione scontrino È inoltre possibile cambiare l'intestazione dello scontrino inserendo informazioni sul ristorante o ciò che si preferisce.

6.5 gestione avvisi e prevenzione errori

segue una lista delle misure di prevenzione che il programma adotta per evitare errori:

- nei field dove devono essere inseriti numeri (per esempio il prezzo del piatto, il numero del tavolo, l'importo ricevuto) è impossibile inserire caratteri che non lo siano (vengono riconosciuti solo numeri da 0 a 9 e il punto "." per inserire i centesimi.)
- nel ruolo di **cameriere** è impossibile selezionare un piatto dal menù se non è stato prima inserito il numero del tavolo così da evitare che vengano inseriti piatti in tavoli che non hanno un numero e che manderebbero in errore il programma.
- nel ruolo di **cameriere** non si possono aggiungere piatti che non sono disponibili nel menù. Viene aperta una schermata di avviso che avvisa della non disponibilità del piatto
- nel ruolo di **cassa** non è possibile generare uno scontrino se l'importo inserito è inferiore a quello del totale dell'anteprima.Viene aperta una schermata di avviso che prega l'utente di inserire un importo sufficiente.
- è impossibile generare uno scontrino se non è stato inserito l'importo dovuto.