



WINE SHOP

By

Andrea Abretti

Alessio Panizzieri

Indice

Introduzione	3
Analisi del testo e specifica dei requisiti	4
Funzionalità del sistema.....	6
Requisiti funzionali	6
Funzionalità.....	7
Utenti	7
Produttore	9
Amministratore.....	9
Impiegato.....	11
Cliente.....	12
Progettazione.....	15
Diagramma casi d'uso.....	15
Diagramma classi.....	15
Server → managedb	15
Server	16
Client.....	16
Utilities	18
Utilities → Models.....	19
Base Dati.....	22
Schema E/R.....	22
Realizzazione DataBase in SQL	23
Istruzioni per l'uso.....	25
Strategia per la realizzazione del progetto.....	25

Introduzione

Come prima cosa viene riportata la traccia del progetto da cui siamo partiti per la realizzazione finale.

“L’obiettivo è sviluppare (usando il linguaggio Java) e documentare un sistema software per la vendita online di bottiglie di vino, utilizzando in modo appropriato le tecniche di riferimento della programmazione orientata agli oggetti. Il sistema realizzato sarà installato presso l’azienda che ha finanziato la realizzazione del sistema. Il sistema interagisce con clienti (persone che vogliono acquistare del vino), impiegati (persone che gestiscono la vendita e la spedizione e ricezione dei prodotti) e un amministratore (persona che controlla il lavoro dei dipendenti e gestisce le attività finanziarie dell’azienda). I clienti sono identificati da: nome, cognome, codice fiscale, indirizzo email, numero telefonico e indirizzo di consegna. Gli impiegati e l’amministratore sono identificati da: nome, cognome, codice fiscale, indirizzo email, numero telefonico e indirizzo di residenza. Per le consegne dei vini ai clienti e per la fornitura dei vini, l’azienda si serve dei servizi di un corriere e di un fornitore. Queste due persone sono identificate da: nome, cognome, indirizzo email, numero telefonico, e codice fiscale e indirizzo della loro azienda. I vini, sono identificati dai seguenti attributi: nome, produttore, provenienza, anno, note tecniche, e i vitigni da cui derivano. Ogni bottiglia ha un prezzo che dipende dalla qualità del vino e dal suo numero di vendite. Per poter utilizzare il sistema, un nuovo cliente deve registrarsi sul sistema inserendo nome utente e password e le altre informazioni richieste dal sistema. Un cliente registrato può fare ricerca dei vini per nome e anno di produzione e acquistare bottiglie di vino dopo un accesso autenticato. Ogni volta che un cliente accede nel sistema, il sistema informa il cliente sulle promozioni in corso. Anche gli impiegati e l’amministratore devono accedere al sistema tramite nome utente e password. e possono modificare la loro password. In aggiunta, l’amministratore ha una password iniziale predefinita che può comunque modificare. Inoltre l’amministratore deve registrare (con nome utente e password) gli altri dipendenti, e resettare e cancellare i loro dati di registrazione quando diventa necessario. Gli impiegati e l’amministratore possono fare la ricerca dei clienti per cognome, dei vini per nome e/o anno di produzione, degli ordini di vendita e di acquisto e delle proposte di acquisto per intervallo di date. Inoltre, l’amministratore deve preparare un report mensile sullo stato dell’azienda (introiti, spese, numero bottiglie vendute e disponibili alla vendita, numero di vendite per i diversi vini, valutazione dei dipendenti). Un cliente può acquistare le bottiglie di vino con confezioni da 1 a 5 bottiglie e da casse da 6 e 12 bottiglie. Ognuna di queste casse contiene solo bottiglie dello stesso tipo di vino. La vendita di casse di vino di 6 e 12 bottiglie è valorizzata da uno sconto del 5 e 10%. La vendita di più di una cassa è valorizzata da un ulteriore sconto del 2 e 3%. Il sistema deve tenere traccia per ogni tipo di vino del numero di bottiglie vendute e di quelle ancora disponibili per la vendita. Un cliente può visualizzare i vini in vendita e selezionare uno o più vini e, per ogni vino, selezionare il numero di bottiglie (default 1). Selezionate le bottiglie, il cliente può decidere di acquistarle. Il sistema permette questa operazione fornendo la possibilità di pagare tramite un bonifico o una carta di credito. Fatto il pagamento, il cliente riceve un ordine di vendita contenente le informazioni su: il cliente che ha fatto l’acquisto, i vini acquistati (nome vino, numero bottiglie e prezzo), l’indirizzo di consegna e la data di consegna. Quando non c’è un numero sufficiente di bottiglie di uno o più vini per soddisfare la richiesta del cliente del numero di bottiglie che vorrebbe acquistare: quando le bottiglie richieste saranno disponibili, il sistema notificherà la loro disponibilità all’utente. Nel caso in cui il numero di bottiglie di uno o più vini non riesce a soddisfare la richiesta del cliente, allora il cliente può compilare una proposta di acquisto contenente: i vini che vuole acquistare (nome vino, numero bottiglie) e l’indirizzo di consegna. Ricevuta la proposta di acquisto, il sistema invia una copia dell’ordine di vendita ad uno degli impiegati. Questo impiegato preparerà un ordine di acquisto e lo invia al fornitore. Ricevuto l’ordine, il fornito prepara il materiale ed effettua la consegna. Ricevuto il materiale dal fornitore, l’impiegato chiede al sistema di generare un ordine di vendita partendo dai dati della proposta di acquisto, dai prezzi dei vini e dalla data di consegna (definita con il corriere) e invia l’ordine di vendita. Ricevuto l’ordine di vendita, il cliente può decidere se acquistarlo o no. Se sì, il cliente deve pagare tramite il sistema; se no, il cliente deve annullare l’acquisto. Il sistema e il dipendente che ha seguito la possibile vendita devono completare le attività necessarie in base alla scelta fatta dal cliente. Il sistema mantiene e gestisce gli ordini di vendita ricevuti dai clienti. In particolare, ogni volta che viene effettuato un acquisto, il sistema aggiorna il numero di vendita e disponibilità dei vini, e invia una copia

dell'ordine di vendita ad uno degli impiegati. Questo impiegato deve esaminare l'ordine, preparare il materiale per la spedizione, concordare con il corriere la data di prelievo e consegna del materiale e informare il cliente sulla data di consegna. Dopo la consegna del materiale al corriere, il dipendente deve completare l'ordine di vendita firmandolo digitalmente. Il sistema deve garantire che per ogni tipo di vino ci sia sempre un numero ragionevole di bottiglie. Per gestire questo vincolo, ad ogni arrivo di un ordine di acquisto il sistema controlla se il numero di bottiglie di alcuni vini è sceso sotto la soglia (il valore della soglia può dipendere dal vino e dai suoi numeri di vendita). Se ciò avviene, il sistema informa uno dei dipendenti sulle quantità disponibili di questi vini. Ricevuta l'informazione, l'impiegato preparerà un ordine di acquisto e lo invia al fornitore. Ricevuto l'ordine, il fornitore prepara il materiale ed effettua la consegna. Ricevuto il materiale dal fornitore, l'impiegato deve completare l'ordine di acquisto firmandolo digitalmente, e caricare l'ordine sul sistema. Il sistema elabora l'ordine e aggiorna le quantità di vini. Il sistema è basato su un'architettura client server. Il server deve supportare l'accesso in parallelo di diversi nodi client. L'interfaccia grafica dei client deve essere implementata con JavaFX. Il database del sistema deve mantenere le informazioni di almeno una decina di vini differenti. Il numero di impiegati al lavoro deve essere almeno tre e il sistema assegna le attività agli impiegati con una politica "round-robin" ovviamente tenendo conto degli impiegati al momento impegnati. Se l'attività non è terminata entro un tempo prestabilito, il sistema l'asigna ad un altro impiegato. Il sistema mantiene le informazioni sulle attività completate e non completate dai dipendenti."

Analisi del testo e specifica dei requisiti

In questa sezione, vengono delineate tutte le entità identificate durante l'analisi del sistema di vendita vini, insieme ai rispettivi attributi e alle funzionalità che il programma incorpora. Durante il processo di analisi, sono stati individuati gli attori, i casi d'uso e le relazioni tra di essi, contribuendo alla definizione complessiva del programma.

È importante sottolineare che ogni attore può sfruttare una o più funzionalità tramite i casi d'uso, e, di conseguenza, un singolo caso d'uso può coinvolgere più attori. Tuttavia, è fondamentale notare che ciascun caso d'uso può avere un solo attore principale.

Il sistema di vendita vini è progettato per coinvolgere diversi tipi di utenti, ognuno dei quali interagisce con il servizio in modi unici. Gli utenti identificati sono il Produttore, l'Amministratore, l'Impiegato e il Cliente, ciascuno dei quali ha accesso a specifiche funzionalità del sistema. Una panoramica dettagliata di ciascun tipo di utente è la seguente:

Produttore:

- **Descrizione:** Rappresenta un ente o una persona responsabile della produzione del vino. Può gestire il catalogo dei vini, fornire dettagli sui nuovi prodotti e interagire con l'azienda di vendita vini per stabilire contratti di fornitura.
- **Casi d'Uso Principali:**
 - **Gestione Catalogo Vini:** Aggiornare le informazioni sui vini prodotti.
 - **Inserire i propri Vini:** Possibilità di inserire ogni volta un nuovo vino.
 - **Eliminazione Vini:** Possibilità di eliminare un vino dal sistema.

Amministratore:

- **Descrizione:** Ha il ruolo di supervisore e gestore dell'intero sistema. Può eseguire attività amministrative, monitorare le performance, e gestire gli utenti e le autorizzazioni.
- **Casi d'Uso Principali:**
 - **Gestione Impiegati:** Aggiungere o eliminare impiegati nel sistema.
 - **Generazione Report:** Creare report mensili sullo stato dell'azienda.
 - **Monitoraggio degli Ordini:** Controllare lo stato degli ordini e controllare il dettaglio dell'ordine selezionato.
 - **Controllare le Offerte:** Monitorare e verificare le offerte presenti nel DB.
 - **Ricerca clienti:** Possibilità di cercare i vari clienti e visualizzarne i dati.

Impiegato:

- **Descrizione:** Responsabile delle attività quotidiane, come la gestione degli ordini, delle spedizioni e della comunicazione con clienti, fornitori e corrieri.
- **Casi d'Uso Principali:**
 - **Gestione Ordini:** Elaborare gli ordini ricevuti dai clienti.
 - **Gestione Spedizioni:** Aggiornare quando necessario lo stato dell'ordine.
 - **Gestione Offerte:** Inserimento e eliminazione di offerte che facciano riferimento a uno specifico vino.
 - **Ricerca clienti:** Possibilità di cercare i vari clienti e visualizzarne i dati.

Cliente:

- **Descrizione:** Rappresenta un acquirente di bottiglie di vino. Può cercare vini, effettuare acquisti, visualizzare e gestire ordini.
- **Casi d'Uso Principali:**
 - **Ricerca e Acquisto Vini:** Effettuare ricerche e acquisti di bottiglie di vino.
 - **Preferiti:** Può aggiungere il vino selezionato in una lista di preferiti.
 - **Recensioni:** Può redigere recensioni su tutti i vini.
 - **Visualizzare il Carrello:** Visualizzare i vini inseriti all'interno del carrello per poi completare l'acquisto inserendo l'indirizzo di spedizione e il metodo di pagamento desiderato.
 - **Assistenza:** Il cliente può richiedere assistenza o redigere una proposta d'acquisto nel caso in cui la quantità del vino desiderato non sia sufficiente, inviando una mail all'impiegato selezionato.
 - **Offerte:** Può visualizzare tutte le offerte disponibili sui vari vini.
 - **Gestione Ordini:** Visualizzare e gestire gli ordini effettuati.
 - **Dettagli Ordine:** Visualizzare tutti i vari vini acquistati in un determinato ordine.

Ogni attore contribuisce al sistema con funzionalità specifiche in base al proprio ruolo e alle proprie responsabilità. I casi d'uso associati ad ogni attore riflettono le azioni principali che possono essere eseguite all'interno del sistema di vendita vini.

Funzionalità del sistema

Requisiti funzionali

Specifica dei casi d'uso per ogni singola entità:

Utenti (Produttore, Amministratore, Impiegato, Cliente)

- Registrazione.
- Login.
- Logout.
- Dati Profilo.
- Ricerca (Clienti, Vini, Ordini, Offerte).
- Mostra (Vino, Cliente, Offerta, Ordine).

Produttore

- Tutte le funzioni di Utenti, a eccezione di Ricerca e Mostra.
- Gestione Vini (Add, Elimina e Modifica).

Amministratore

- Tutte le funzioni di Utenti.
- Gestione Impiegato (Add e Elimina).
- Report mensile.
- Gestione Ordini (Visualizzazione Ordine).
- Visualizza Dettaglio Ordine.
- Gestione Offerte (Elimina e Visualizza Offerta).

Impiegato

- Tutte le funzioni di Utenti, a eccezione della Registrazione.
- Gestione Ordini (Modifica Stato Ordine).
- Gestione Offerte (Add, Elimina e Modifica).
- Gestione Assistenze.

Cliente

- Tutte le funzioni di Utenti.
- Add Carrello.

- Gestione Recensioni.
- Gestione Preferiti (Add e Elimina).
- Gestione Ordini (Visualizza).
- Visualizza Dettaglio Ordine.
- Richiedi Assistenza.

Funzionalità

Utenti

Registrazione	
Descrizione	Permette ad un utente di registrarsi all'interno del DataBase, specificando anche il ruolo.
Precondizioni	Compilare tutti i campi, rispettando le condizioni (Es. numero telefonico di 10 numeri).
Frequenza d'uso	Ogni qual volta un utente si voglia registrare al sistema.
Esecuzione	Controllo dell'effettiva validità dei dati inseriti e successivamente inserisce l'utente nel sistema.

Login	
Descrizione	Permette agli utenti di autenticarsi nel sistema, inserendo username e password.
Precondizioni	L'utente che tenta di autenticarsi deve essersi registrato precedentemente e di conseguenza presente nel DB.
Frequenza d'uso	Ogni qual volta un utente vuole accedere al sistema.
Esecuzione	Inserire Username e Password e di conseguenza il sistema controllerà se coincidono con quelli di qualche utente presente nel DB.

Logout	
Descrizione	Permette all'utente autenticato di disconnettersi dal sistema, senza lasciare il proprio profilo aperto.
Precondizioni	L'utente deve avere svolto l'accesso al sistema precedentemente.
Frequenza d'uso	Ogni qual volta ci si voglia disconnettere.
Esecuzione	Premere il bottone di "Logout" e di conseguenza disconnettersi dal sistema

Dati Profilo	
Descrizione	Permette di visualizzare i propri dati come (Nome, Cognome, Indirizzo, Email, Telefono ecc...)
Precondizioni	L'utente deve essere collegato al sistema e deve aver svolto l'accesso.
Frequenza d'uso	Ogni qual volta si vogliono visualizzare i propri dati.
Esecuzione	Premere il bottone "Profilo" per visualizzare i dati.

Ricerca	
Descrizione	Si intendono tutte le tipologie di ricerca (Vini, Clienti, Impiegati, Offerte e Ordini). Permette all'utente loggato di ricercare un dato tramite una barra di ricerca.
Precondizioni	Il dato cercato deve essere presente all'interno del sistema.
Frequenza d'uso	Ogni qual volta si vuole ricercare un determinato dato.
Esecuzione	Inserire il dato da cercare nella barra di ricerca e successivamente premere il bottone a fianco.

Mostra	
Descrizione	Si intendono tutte le tipologie di Mostra (Vini, Clienti, Impiegati, Offerte e Ordini). Permette di visualizzare i dati dell'elemento selezionato.
Precondizioni	L'elemento deve essere presente nel DB e selezionabile per poterlo visualizzare.
Frequenza d'uso	Ogni qual volta si vogliono visualizzare i dati dell'elemento selezionato.
Esecuzione	Premere sull'elemento che si vuole vedere per visualizzare i suoi dati.

Produttore

Gestione Vini	
Descrizione	Sono compresi tutti le funzionalità di gestione come: Add, Elimina e Modifica. Permette di andare a lavorare direttamente sui vini, in modo da inserirli/modificarli/eliminarli.
Precondizioni	L'utente che vuole gestire i vini deve avere i privilegi per farlo, ovvero deve essere un: Produttore.
Frequenza d'uso	Ogni qual volta si voglia o si debba andare a lavorare sui dati dei vini.
Esecuzione	Premere sul bottone "Gestione" del vino selezionato per modificarlo/eliminarlo oppure premere su "Inserisci vino" per andare ad aggiungere un vino.

Amministratore

Gestione Impiegato	
Descrizione	Sono compresi tutti le funzionalità di gestione come: Add e Elimina. Permette di andare gestire questi due casi sugli impiegati, lavorando sui suoi dati.
Precondizioni	Per gestire gli impiegati, l'utente loggato deve essere registrato come Amministratore.

Frequenza d'uso	Ogni qual volta si voglia aggiungere o eliminare un impiegato
Esecuzione	Premere sul bottone “Aggiunta impiegati” per inserire, o premere su gestione nella TableView di Ricerca Impiegati per poterlo eliminare.

Report Mensile	
Descrizione	L'amministratore deve redigere un report mensile sull'andamento dell'azienda.
Precondizioni	L'utente per redigere un report deve essere classificato come “Amministratore”.
Frequenza d'uso	Una volta al mese per tenere conto dell'andamento dell'azienda.
Esecuzione	Selezionare la data in cui viene redatto il report, di seguito digitare la descrizione del report e successivamente aggiungerlo al DB.

Gestione Ordini	
Descrizione	Permette di visualizzare i dati di un ordine e il suo stato in quell'istante preciso.
Precondizioni	Nessuna precondizione particolare.
Frequenza d'uso	Ogni qual volta si vogliano visualizzare i dati di un determinato ordine.
Esecuzione	Nella TableView di Ricerca Ordini selezionare l'ordine che si vuole visualizzare.

Visualizza Dettaglio Ordine	
Descrizione	Permette di visualizzare i dettagli di un ordine, ovvero vedere nello specifico tutti i vini presenti in quell'ordine con la relativa quantità.
Precondizioni	Nessuna precondizione particolare
Frequenza d'uso	Ogni qual volta si voglia visualizzare nello specifico i vini e la quantità acquistata.

Esecuzione	Una volta selezionato l'ordine, premere su "Visualizza dettagli ordine".
------------	--

Gestione Offerte	
Descrizione	Permette di visualizzare le offerte presenti nel DB ed eventualmente eliminarle.
Precondizioni	L'utente che può svolgere azioni sulle offerte sono: Amministratori o Impiegati.
Frequenza d'uso	Ogni qual volta si vogliano visualizzare i dettagli di un'offerta o nel caso in cui la si voglia eliminare.
Esecuzione	Selezionare l'offerta che si desidera visualizzare e gestire.

Impiegato

Gestione Ordini	
Descrizione	Permette di visualizzare i dati di un ordine, il suo stato in quell'istante preciso e di modificarlo quando necessario. Naturalmente una volta che lo stato viene cambiato in "Consegnato" non è più possibile modificarlo nuovamente.
Precondizioni	L'utente loggato deve essere un "Impiegato".
Frequenza d'uso	Ogni qual volta si vogliano visualizzare i dati di un determinato ordine o modificare il suo stato.
Esecuzione	Una volta selezionato l'ordine da visualizzare sarà possibile modificare il suo stato a meno che non sia già "Consegnato".

Gestione Offerte	
Descrizione	Permette di aggiungere, visualizzare ed eliminare offerte
Precondizioni	L'utente che può svolgere azioni sulle offerte sono: Amministratori o Impiegati.

Frequenza d'uso	Ogni qual volta si vogliano aggiungere, visualizzare o eliminare un'offerta.
Esecuzione	Selezionare l'offerta che si desidera visualizzare/gestire oppure andare nella sezione di inserimento offerte.

Gestione Assistenza	
Descrizione	Permette di visualizzare tutte le richieste di assistenza che sono arrivate tramite mail.
Precondizioni	L'utente loggato deve essere di tipo "Impiegato".
Frequenza d'uso	Ogni qual volta arrivi una nuova richiesta e la si voglia visualizzare.
Esecuzione	Andare nella sezione "Assistenze" e selezionare la richiesta che si vuole visualizzare.

Cliente

Add Carrello	
Descrizione	Dopo aver inserito la quantità desiderata permette di inserire il vino selezionato nel carrello.
Precondizioni	Dopo aver selezionato la quantità desiderata il vino deve essere ancora disponibile, ovvero che la quantità rimanente non deve essere inferiore alla soglia.
Frequenza d'uso	Ogni qual volta si voglia aggiungere un vino al carrello.
Esecuzione	Selezionare la quantità e premere il bottone "Aggiungi al carrello".

Gestione Recensioni	
Descrizione	Permette di aggiungere o eliminare una recensione per il vino selezionato.

Precondizioni	L'utente loggato deve essere di tipo "Cliente", mentre per l'eliminazione è necessario che vi sia almeno una recensione nella lista.
Frequenza d'uso	Ogni qual volta si voglia effettuare una recensione su un determinato vino o nel caso in cui la si voglia eliminare.
Esecuzione	Per aggiungere una recensione è necessario selezionare un vino, inserire un voto da 1 a 5 e successivamente scrivere la recensione. Mentre per l'eliminazione bisogna andare nella sezione "Recensioni" e selezionare la recensione da eliminare.

Gestione preferiti	
Descrizione	Permette di aggiungere un vino ad una lista di preferiti o di toglierlo.
Precondizioni	L'utente loggato deve essere un "Cliente", mentre per l'eliminazione è necessario che vi sia almeno un vino nella lista dei preferiti.
Frequenza d'uso	Ogni qual volta si acquista un vino che è piaciuto molto e di conseguenza lo si vuole memorizzare da qualche parte.
Esecuzione	Nella scheda del vino selezionato, premere il bottone "Aggiungi preferiti".

Gestione Ordini	
Descrizione	Permette di visualizzare i dati di un ordine e il suo stato in quell'istante preciso.
Precondizioni	Nessuna precondizione particolare.
Frequenza d'uso	Ogni qual volta si vogliano visualizzare i dati di un determinato ordine.
Esecuzione	Nella TableView di Ricerca Ordini selezionare l'ordine che si vuole visualizzare.

Visualizza Dettaglio Ordine	
Descrizione	Permette di visualizzare i dettagli di un ordine, ovvero vedere nello specifico tutti i vini presenti in quell'ordine con la relativa quantità.
Precondizioni	Nessuna precondizione particolare
Frequenza d'uso	Ogni qual volta si voglia visualizzare nello specifico i vini e la quantità acquistata.
Esecuzione	Una volta selezionato l'ordine, premere su "Visualizza dettagli ordine".

Richiedi Assistenza	
Descrizione	Permette di selezionare il dipendente che si vuole contattare e redigere una email (Oggetto e Testo) dove si esprime una problematica da risolvere o una proposta d'acquisto nel caso in cui la quantità di un determinato vino che si intende acquistare sia troppa.
Precondizioni	L'utente loggato deve essere un "Cliente" e deve possedere un account gmail.
Frequenza d'uso	Ogni qual volta si necessita di assistenza da parte dell'azienda WineShop.
Esecuzione	Selezionare l'impiegato da contattare, inserire l'oggetto, il testo e successivamente premere il bottone "Invia email".

Progettazione

Diagramma casi d'uso

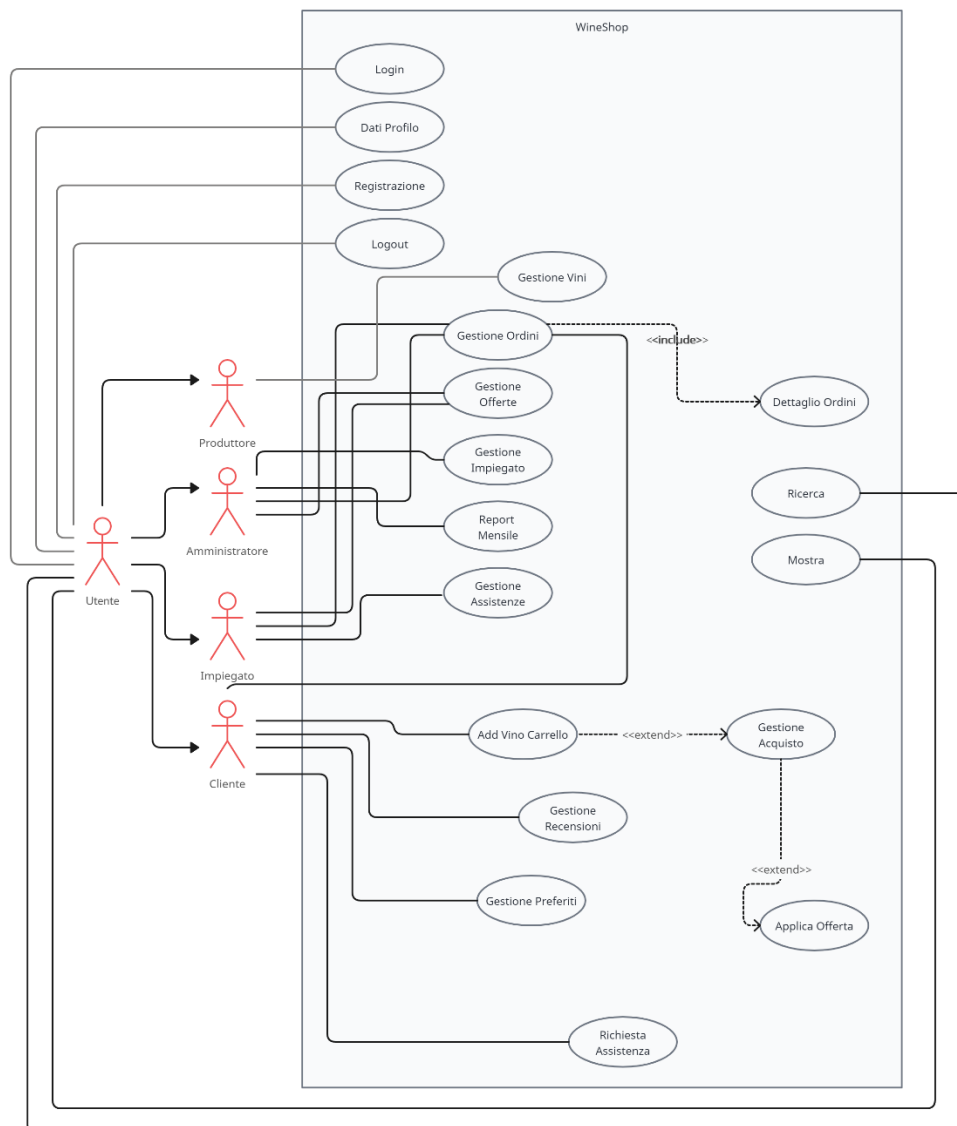










Diagramma classi

Server → managedb






















  SQLTranslator
  SQLTranslator()   requestToSQL(Request): String   sqlToResponse(List<Map<String, String>>): Response








Server









  ClientHandler
  ClientHandler(Socket, ServerConfiguration)   run(): void















  Server
  main(String[]): void





















  ServerConfiguration
  ServerConfiguration(String)   getDbHost(): String   getServerPort(): int   getDbUser(): String   getDbPassword(): String   getDbName(): String   getDbPort(): int   toString(): String  Object

Client

  Client
  start(Stage): void  Application   main(String[]): void

  RequestController
  RequestController(ObjectOutputStream, ObjectInputStream)   makeRequest(int, T): Response   closeConnection(): void

  LoginController
  setRequestController(RequestController): void   setUtenteLoggato(Utente): void   OnLogin_Login_BtnLogin(ActionEvent): void   OnLogin_MostraClick(ActionEvent): void   OnLogin_BtnRegClick(ActionEvent): void   OnLogin_LinkClick(ActionEvent): void





























  RegistrazioneController
  setRequestController(RequestController): void   CF_Check(String): boolean   initialize(URL, ResourceBundle): void   OnReg_BtnLoginClick(ActionEvent): void   OnReg_BtnRegClick(ActionEvent): void   OnReg_MostraClick(ActionEvent): void   Errore_Campi_Vuoti(String): void   Errore_Email(): void   Errore_Password(): void

C HomeProduttoreController	
setRequestController(RequestController): void	
setLoggedUser(Utente): void	
initialize(): void	
OnBtnCaricaImag_Click(ActionEvent): void	
OnBtnGestioneViniProduttore_Click(ActionEvent): void	
OnBtnHomeProduttore_Click(ActionEvent): void	
OnBtnInsVinoProduttore_Click(ActionEvent): void	
OnBtnInserisciVino_Click(ActionEvent): void	
OnBtnLogoutProduttore_Click(ActionEvent): void	
OnBtnProfiloProduttore_Click(ActionEvent): void	
OnBtnRicercaViniProduttore_Click(ActionEvent): void	
OnBtnModificaImag_Click(ActionEvent): void	
OnBtnModificaVino_Click(ActionEvent): void	
OnBtnDeleteVino_Click(ActionEvent): void	
Errore_Anno(): void	
Errore_Campi_Vuoti(String): void	
Errore_Interi(): void	
Errore_Numeri(): void	






C HomeImpiegatoController	
setRequestController(RequestController): void	
setLoggedUser(Utente): void	
initialize(): void	
OnBtnHomeImpiegato_Click(ActionEvent): void	
OnBtnLogoutImpiegato_Click(ActionEvent): void	
OnBtnProfiloImpiegato_Click(ActionEvent): void	
OnBtnRicercaImpiegato_Click(ActionEvent): void	
OnBtnRicerca_Click(ActionEvent): void	
OnBtnApplicaOffertaVino_Click(ActionEvent): void	
OnBtnOffertaImpiegato_Click(ActionEvent): void	
OnBtnCercaOfferte_Click(ActionEvent): void	
OnBtnInserisciOfferta_Click(ActionEvent): void	
OnBtnEliminaOfferta_Click(ActionEvent): void	
OnBtnModificaOfferta_Click(ActionEvent): void	
OnBtnOrdiniImpiegato_Click(ActionEvent): void	
OnBtnCercaOrdini_Click(ActionEvent): void	
OnBtnModificaStatoOrdine_Click(ActionEvent): void	
OnBtnVisualizzaDettagliOrdine_Click(ActionEvent): void	
OnBtnAssistenzaImpiegato_Click(ActionEvent): void	
Errore_Campi_Vuoti(String): void	
Errore_Sconto(): void	





C AggiornaController	
setRequestController(RequestController): void	
setLoggedUser(Utente): void	
initialize(): void	
OnCambioPass_BtnIndietro_Click(ActionEvent): void	
OnCambioPass_Mostra_Click(ActionEvent): void	
OnCambioPass_Click(ActionEvent): void	
Errore_Password(): void	
Errore_Campi_Vuoti(String): void	

C HomeAmministratoreController	
setRequestController(RequestController): void	
setLoggedUser(Utente): void	
CF_Check(String): boolean	
initialize(): void	
OnBtnHomeAmministratore_Click(ActionEvent): void	
OnBtnLogoutAmministratore_Click(ActionEvent): void	
OnBtnAddImpiegatiAmministratore_Click(ActionEvent): void	
OnBtnAddImpiegati_Click(ActionEvent): void	
OnBtnAddReport_Click(ActionEvent): void	
OnBtnOrdiniAmministratore_Click(ActionEvent): void	
OnBtnCercaOrdini_Click(ActionEvent): void	
OnBtnVisualizzaDettagliOrdine_Click(ActionEvent): void	
OnBtnProfiloAmministratore_Click(ActionEvent): void	
OnBtnReportAmministratore_Click(ActionEvent): void	
OnBtnRicercaAmministratore_Click(ActionEvent): void	
OnBtnRicerca_Click(ActionEvent): void	
OnBtnEliminaImpiegato_Click(ActionEvent): void	
OnBtnEliminaOfferta_Click(ActionEvent): void	
Errore_Campi_Vuoti(String): void	
Errore_Email(): void	
Errore_Password(): void	






C  HomeClienteController	
m 	setRequestController(RequestController): void
m 	setLoggedUser(Utente): void
m 	OnBtnLogoutCliente_Click(ActionEvent): void
m 	initialize(): void
m 	OnBtnHomeCliente_Click(ActionEvent): void
m 	OnBtnRicercaCliente_Click(ActionEvent): void
m 	OnBtnRicercaViniCliente_Click(ActionEvent): void
m 	OnBtnAggiungiCarrelloVino_Click(ActionEvent): void
m 	OnBtnCarrelloCliente_Click(ActionEvent): void
m 	OnBtnAcquistaCarrello_Click(ActionEvent): void
m 	OnBtnPromozioniCliente_Click(ActionEvent): void
m 	OnBtnCercaOfferte_Click(ActionEvent): void
m 	OnBtnOrdiniCliente_Click(ActionEvent): void
m 	OnBtnCercaOrdini_Click(ActionEvent): void
m 	OnBtnPreferitiCliente_Click(ActionEvent): void
m 	OnBtnRicercaPreferiti_Click(ActionEvent): void
m 	OnBtnRecensioneCliente_Click(ActionEvent): void
m 	OnBtnRicercaRecensioni_Click(ActionEvent): void
m 	OnBtnAddRecensione_Click(ActionEvent): void
m 	OnBtnPreferitiVino_Click(ActionEvent): void
m 	OnBtnVisualizzaDettagliOrdine_Click(ActionEvent): void
m 	OnBtnAssistenzaCliente_Click(ActionEvent): void
m 	OnBtnInviaEmail_Click(ActionEvent): void
m 	OnBtnProfiloCliente_Click(ActionEvent): void
m 	Errore_Campi_Vuoti(): void
m 	Calcolo_Sconto(double, double): double
m 	sendEmail(String, String, String): void

Utilities

C  Response	
m 	Response(int, Object)
m 	getStatusCode(): int
m 	getPayloadType(): Class<?>
m 	getPayload(): Object












































C  Request	
m 	Request(int, Object)
m 	getValue(): int
m 	getPayloadType(): Class<?>
m 	getPayload(): Object






























































C  EmptyPayload	
m 	EmptyPayload()
m 	EmptyPayload(String)
m 	toString(): String ↑Object

I  Insertable	
m 	getAttributes(): String[]
m 	getInstanceName(): String
m 	getValues(): String[]
m 	getPk(): String





























































I  Removable	
m 	getPkValue(): String











































Utilities → Models

 Assistenza
  Assistenza(int, int, int, String)
  Assistenza(int, int, String)
  getId(): int
  setId(int): void
  getCODImpiegato(): int
  setCODImpiegato(int): void
  getCODCliente(): int
  setCODCliente(int): void
  getProposta_Acquisto(): String
  setProposta_Acquisto(String): void
  getAttributes(): String[]  
  getInstanceName(): String  
  getValues(): String[]  
  getPk(): String  
  getPkValue(): String
  toString(): String  

 DettagliOrdine
  DettagliOrdine(int, int, int, int)
  DettagliOrdine(int)
  DettagliOrdine(int, int, int)
  DettagliOrdine(double, int, int)
  DettagliOrdine(int, int)
  DettagliOrdine(int, int, int, String)
  DettagliOrdine(int, int, String)
  getCODCliente(): int
  setCODCliente(int): void
  getId(): int
  setId(int): void
  getQuantita(): int
  setQuantita(int): void
  getCODOrdine(): int
  setCODOrdine(int): void
  getCODVino(): int
  setCODVino(int): void
  getNome_vino(): String
  setNome_vino(String): void
  getAttributes(): String[]  
  getInstanceName(): String  
  getValues(): String[]  
  getPk(): String  
  getPkValue(): String
  toString(): String  

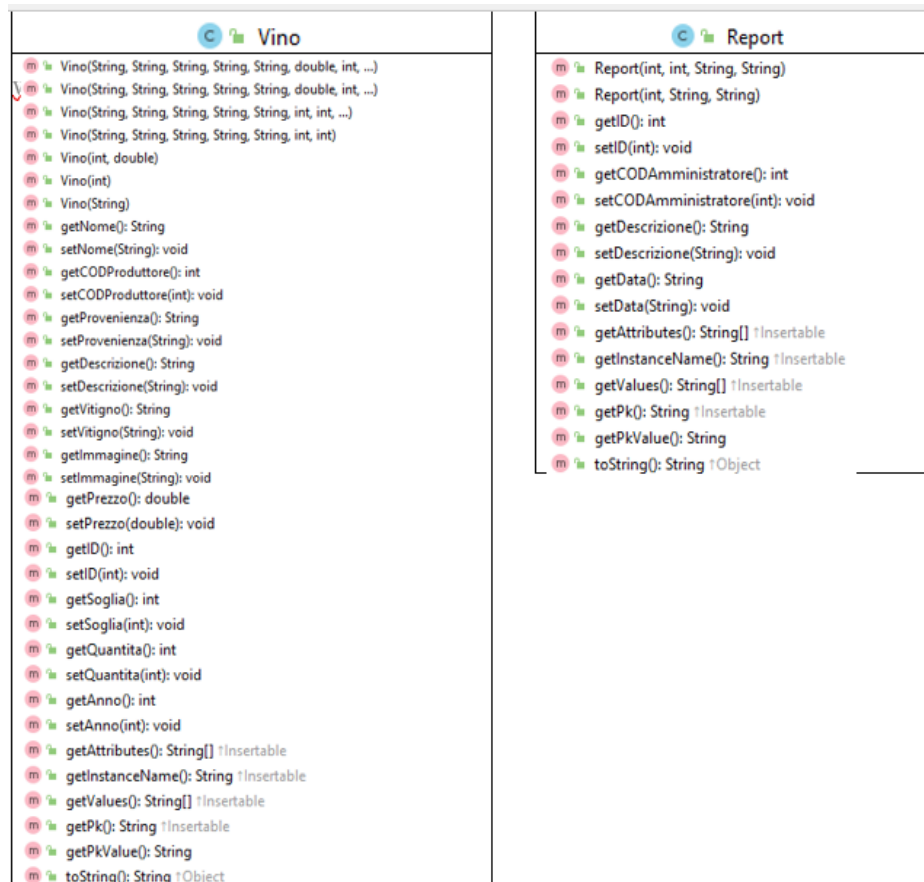
 Offerta
  Offerta(int, int, int, String)
  Offerta(int)
  Offerta(double)
  Offerta(int, int)
  Offerta(int, int, String)
  Offerta(int, int, String, String)
  getId(): int
  setId(int): void
  getCODVino(): int
  setCODVino(int): void
  getSconto(): int
  setSconto(int): void
  getDescrizione(): String
  setDescription(String): void
  getNome(): String
  setNome(String): void
  toString(): String  
  getAttributes(): String[]  
  getInstanceName(): String  
  getValues(): String[]  
  getPk(): String  
  getPkValue(): String

 Ordine
  Ordine(int, double, int, String, String, String)
  Ordine(int, double, String, String)
  Ordine(int, String, String)
  Ordine(String)
  Ordine(int, String)
  Ordine(int)
  Ordine(double)
  Ordine(String, int)
  Ordine(double, String)
  getId(): int
  setId(int): void
  getTotale(): double
  setTotale(double): void
  getCODCliente(): int
  setCODCliente(int): void
  getMetodo_Pagamento(): String
  setMetodo_Pagamento(String): void
  getIndirizzo(): String
  setIndirizzo(String): void
  getStato(): String
  setStato(String): void
  getAttributes(): String[] 
  getInstanceName(): String 
  getValues(): String[] 
  getPk(): String 
  getPkValue(): String
  toString(): String 

 Preferito
  Preferito(int, int, int)
  Preferito(int, int)
  Preferito(int)
  Preferito(int, int, int, String)
  getId(): int
  setId(int): void
  getCODVino(): int
  setCODVino(int): void
  getCODCliente(): int
  setCODCliente(int): void
  getNome(): String
  setNome(String): void
  getAttributes(): String[] 
  getInstanceName(): String 
  getValues(): String[] 
  getPk(): String 
  getPkValue(): String
  toString(): String 

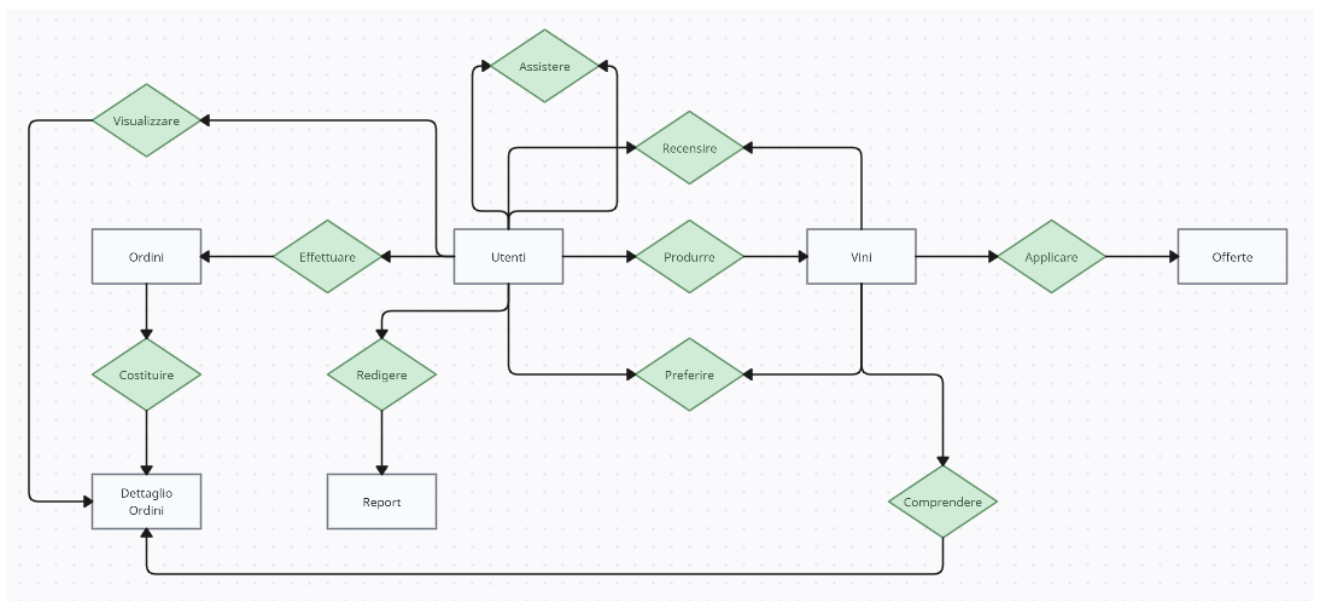
Recensione	
m	Recensione(int, String, int, int, String)
m	Recensione(int, String, int, String)
m	Recensione(int, int, String, String)
m	Recensione(int)
m	Recensione(int, int)
m	Recensione(int, String)
m	getID(): int
m	setID(int): void
m	getVoto(): String
m	setVoto(String): void
m	getCODVino(): int
m	setCODVino(int): void
m	getCODCliente(): int
m	setCODCliente(int): void
m	getCommento(): String
m	setCommento(String): void
m	getAttributes(): String[] †Insertable
m	getInstanceName(): String †Insertable
m	getValues(): String[] †Insertable
m	getPk(): String †Insertable
m	getPkValue(): String
m	toString(): String †Object

Utente	
m	Utente(String, String)
m	Utente(int, String, String, String)
m	Utente(String, int)
m	Utente(String, String, String, String, String, String, String, String, ...)
m	Utente(String, String, String, String, String, String, String, ...)
m	Utente(String, String, String, String, String, String, String, ...)
m	Utente(int)
m	Utente(String)
m	Utente(String, boolean)
m	getNome(): String
m	setNome(String): void
m	getCognome(): String
m	setCognome(String): void
m	getCf(): String
m	setCf(String): void
m	getEmail(): String
m	setEmail(String): void
m	getTelefono(): String
m	setTelefono(String): void
m	getIndirizzo(): String
m	setIndirizzo(String): void
m	getPassword(): String
m	setPassword(String): void
m	getTipo(): String
m	setTipo(String): void
m	getUsername(): String
m	setUsername(String): void
m	getId(): int
m	setId(int): void
m	getAttributes(): String[] †Insertable
m	getInstanceName(): String †Insertable
m	getValues(): String[] †Insertable
m	getPk(): String †Insertable
m	getPkValue(): String
m	toString(): String †Object



Base Dati

Schema E/R



Realizzazione DataBase in SQL

```
CREATE DATABASE wine_shop;

CREATE TABLE utenti(
    ID int AUTO_INCREMENT NOT null PRIMARY KEY,
    CF varchar(17),
    Nome varchar(10),
    Cognome varchar(10),
    Username varchar(10) UNIQUE,
    Password varchar(50),
    Email varchar(30),
    Telefono varchar(10),
    Indirizzo varchar(50),
    Tipo varchar(20),
    Online tinyint(1),
    LastLogin datetime
);

CREATE TABLE vini(
    ID int AUTO_INCREMENT NOT null PRIMARY KEY,
    Nome varchar(100),
    Provenienza varchar(50),
    Anno int,
    Descrizione varchar(1000),
    Vitigno varchar(50),
    Prezzo double,
    Soglia int,
    Quantita int,
    Immagine varchar(1000),
    CODProduttore int,
    FOREIGN KEY(CODProduttore) REFERENCES utenti(ID)
);

CREATE TABLE report(
    ID int AUTO_INCREMENT NOT null PRIMARY KEY,
    Descrizione varchar(500),
    Data varchar(15),
    CODAmministratore int,
    FOREIGN KEY(CODAmministratore) REFERENCES utenti(ID)
);

CREATE TABLE recensioni(
    ID int AUTO_INCREMENT NOT null PRIMARY KEY,
    Voto varchar(1),
    Commento varchar(1000),
    CODVino int,
    CODCliente int,
    FOREIGN KEY(CODVino) REFERENCES vini(ID),
    FOREIGN KEY(CODCliente) REFERENCES utenti(ID)
);
```

```
CREATE TABLE preferiti(  
    ID int AUTO_INCREMENT NOT null PRIMARY KEY,  
    CODVino int,  
    CODCliente int,  
    FOREIGN KEY(CODVino) REFERENCES vini(ID),  
    FOREIGN KEY(CODCliente) REFERENCES utenti(ID)  
);
```

```
CREATE TABLE ordini(  
    ID int AUTO_INCREMENT NOT null PRIMARY KEY,  
    Totale double,  
    Indirizzo varchar(50),  
    Metodo_Pagamento varchar(20),  
    Stato varchar(20),  
    CODCliente int,  
    FOREIGN KEY(CODCliente) REFERENCES utenti(ID)  
);
```

```
CREATE TABLE offerte(  
    ID int AUTO_INCREMENT NOT null PRIMARY KEY,  
    Descrizione varchar(255),  
    Sconto int,  
    CODVino int,  
    FOREIGN KEY(CODVino) REFERENCES vini(ID)  
);
```

```
CREATE TABLE dettagli_ordini(  
    ID int AUTO_INCREMENT NOT null PRIMARY KEY,  
    Quantita int,  
    CODVino int,  
    CODOrdine int,  
    CODCliente int,  
    FOREIGN KEY(CODOrdine) REFERENCES ordini(ID),  
    FOREIGN KEY(CODVino) REFERENCES vini(ID),  
    FOREIGN KEY(CODCliente) REFERENCES utenti(ID)  
);
```

```
CREATE TABLE assistenze(  
    ID int AUTO_INCREMENT NOT null PRIMARY KEY,  
    Proposta_Acquisto varchar(400),  
    CODImpiegato int,  
    CODCliente int,  
    FOREIGN KEY(CODImpiegato) REFERENCES utenti(ID),  
    FOREIGN KEY(CODCliente) REFERENCES utenti(ID)  
);
```


Istruzioni per l'uso

All'interno della cartella del progetto è presente un file che consente di inserire dati nel database, nelle rispettive cartelle di 'utenti' e 'vini'. Al fine di compilare ed eseguire correttamente il programma, è necessario apportare alcune modifiche. Sul lato server, nella classe **“DataBaseManager.java”** è richiesto di modificare il percorso della stringa alla riga 47. Lo stesso vale per la classe **“Server.java”** dove è necessario inserire la stringa corrispondente alla riga 11. Per quanto riguarda il lato client, nella classe **“Client.java”**, alla riga 71 è necessario modificare la stringa per visualizzare il logo dell'app come icona nella barra degli strumenti in basso. Si noti che i file 'ServerConfig.json' e 'Logo_Calice.png' sono presenti nella cartella del progetto.

Inoltre, è essenziale modificare i percorsi relativi alle immagini nei comandi **INSERT INTO** per i dati dei vini presenti nel file **“wine_shop.sql”**. Si consiglia di inserire i percorsi corrispondenti in modo che le immagini vengano caricate correttamente nel DataBase.

Per quanto riguarda la verifica del funzionamento della posta elettronica per le richieste di assistenza o proposte d'acquisto, di seguito sono forniti gli account Gmail associati all'account Impiegato (Paola) e Cliente (Andrea).

Account Cliente: andreawineshop@gmail.com

Password: wineshop@@.

Account Impiegata: paolaberlini74@gmail.com

Password: 60paola@@.

Tuttavia, desidero sottolineare che l'accesso a questi account è protetto da un'autenticazione a due fattori, che richiede un secondo passo di verifica attraverso un codice inviato direttamente al mio cellulare. Pertanto, eventualmente, la verifica di questa funzionalità di posta elettronica potrebbe essere effettuata in sede d'esame.

Strategia per la realizzazione del progetto

Inizialmente, abbiamo condotto un'analisi approfondita della traccia per comprendere come strutturare la base del progetto, definendo entità, attributi e le varie funzionalità coinvolte. Una volta definiti questi aspetti, abbiamo suddiviso i compiti tra di noi. Durante lo sviluppo del programma, ci tenevamo costantemente aggiornati reciprocamente, fornendoci feedback sulle nuove implementazioni.

L'approccio collaborativo ha coinvolto una comunicazione continua: quando uno di noi apportava modifiche al codice, l'altro veniva prontamente informato, mantenendo un flusso costante di aggiornamenti. Affrontavamo i problemi nel codice in maniera collaborativa, chiedendo aiuto quando

necessario per risolvere le sfide incontrate. Questo approccio non solo favoriva una soluzione più rapida, ma contribuiva anche a mantenere una coerenza nel progresso del progetto.

È importante sottolineare che la maggior parte della collaborazione è avvenuta a distanza, utilizzando GitHub come piattaforma principale. La creazione di un repository ci ha consentito di caricare e condividere le modifiche in modo efficiente, garantendo che entrambi gli sviluppatori fossero sempre informati sullo stato corrente del progetto. Questa modalità di lavoro remota ha dimostrato di essere produttivo nel mantenere un'efficace sincronia tra i membri del team durante lo sviluppo del progetto.

Per lo sviluppo del progetto, abbiamo adottato l'IDE (Integrated Development Environment) IntelliJ IDEA Ultimate fornito da JetBrains, utilizzando una licenza studentesca e programmando in linguaggio Java. Per quanto riguarda la gestione del database, abbiamo impiegato MySQL come tecnologia. La parte grafica è stata implementata attraverso l'utilizzo di Scene Builder, una piattaforma dedicata alla creazione dei diversi form del nostro progetto.