

Ristorante

Gaetano Romeo - Vincenzo Figliolino - Domenico Riso

Gennaio 2024

Descrizione

Si vuole realizzare un sistema informatico per simulare la gestione di un ristorante.

Il sistema deve consentire ad un cliente di comunicare il numero di posti necessari, di effettuare varie ordinazioni e di richiedere il conto.

Inoltre, lo chef deve poter personalizzare il menù a proprio piacimento.

I clienti che vogliono utilizzare il sistema, devono prima completare una fase di registrazione mediante una username ed una password.

Nel loro utilizzo saranno assistiti da un receptionist, che avrà il compito di verificare se ci sono posti disponibili, e dai camerieri, che prenderanno le ordinazioni e forniranno loro i piatti una volta pronti.

Requisiti Tecnici

La scelta del linguaggio di programmazione è ricaduta su Java ed il progetto sarà stilizzato utilizzando JavaFX per permettere agli utenti un facile utilizzo del sistema mediante delle apposite interfacce grafiche.

Per gestire le richieste di accesso o registrazione da parte dei clienti, la memorizzazione delle loro credenziali e del menù in maniera persistente, verrà utilizzato un database MySQL mantenuto in locale.

La gestione dei compiti di camerieri e receptionist sarà affidata al sistema ed il cliente ne sarà all'oscuro.

Struttura: Classi, Controller ed Interfacce

I quattro attori principali del sistema, ovvero il cliente, il receptionist, il cameriere e lo chef, saranno identificati mediante una classe ed eseguiti come processi.

Le classi Customer e Chef, che estenderanno la classe Application (main dell'interfaccia) fungeranno da entry point per le rispettive interfacce grafiche.

Al loro interno verranno definite le modalità di visualizzazione.

Esse avranno delle classi Controller che si occuperanno del loro funzionamento interno in termini di codice, come comunicazioni ed altre operazioni.

Ogni interfaccia sarà gestita e stilizzata mediante un apposito file FXML, all'interno del quale saranno dichiarati gli elementi che la compongono come pulsanti e form di testo.

Per la gestione degli ordini in termini di relazione nome-prezzo, sarà realizzata una classe Order, con rispettivi metodi e costruttore.

La classe Order sarà utilizzata per gestire gli ordini che fanno parte del menù come oggetti con legame nome-prezzo.

A differenza delle altre classi, essa avrà un costruttore di default per l'inizializzazione.

Per la creazione del database, è presente un apposito script SQL.

Le classi che avranno bisogno di accedere al database sono CustomerController per le fasi di accesso e registrazione e ChefController per la fase di modifica del menù.

Schema dell'Architettura

L'intero sistema si baserà su di un paradigma ibrido, a metà tra un client-server ed un peer-to-peer, ed in particolare:

- Il processo Customer fungerà da client, in quanto dovrà prima comunicare al receptionist il numero di posti di cui necessita e poi al cameriere gli ordini desiderati ed il conto
- Il processo Receptionist fungerà da server, in quanto si metterà in attesa di richieste da parte di clienti e fornirà una risposta al cliente in base alle disponibilità del ristorante
- Il processo Waiter fungerà sia da server, in quanto si metterà in attesa di richieste di ordini da parte dei clienti, sia da client, in quanto ne richiederà la preparazione allo chef
- Il processo Chef fungerà da server, in quanto si metterà in attesa di richieste di preparazione di ordini da parte dei camerieri e restituirà gli ordini una volta pronti

Per far sì che i server siano concorrenti e rispondano cioè a più richieste in simultanea, essi genereranno un nuovo thread per gestire ogni richiesta.

Comunicazione

Ogni entità avrà una porta associata, sulla quale comunicherà con un'altra, ed una socket, mediante la quale leggerà o invierà dati.

Il numero di porta dovrà necessariamente essere superiore a 1023 per non rischiare di utilizzare una porta associata ad un servizio del sistema operativo.

La comunicazione avverrà in locale e le socket utilizzate saranno basate sul protocollo TCP, garantendo sicurezza nell'invio e nella ricezione dei dati e il corretto ordine di ricezione.

I client utilizzeranno una socket e specificheranno il proprio indirizzo (in questo caso "localhost") e la porta sulla quale il processo server da raggiungere è in ascolto.

I server utilizzeranno una server socket e specificheranno la porta sulla quale sono in ascolto.

Le primitive utilizzate per la comunicazione saranno bloccanti: quando un client effettuerà una lettura, bloccherà il suo flusso di esecuzione fino a leggere effettivamente i dati richiesti.

Siccome le socket sono bidirezionali (permettono sia operazioni di lettura che di scrittura su entrambi gli estremi), per ogni comunicazione ne sarà sufficiente una.

Comunicazione cliente-receptionist (GetRequiredSeats)

Il cliente vuole comunicare con il receptionist per richiedere dei posti.

La classe Customer dichiara una socket utilizzando un oggetto della classe Socket di Java, specificando l'indirizzo del processo Customer (localhost) e la porta sulla quale il processo Receptionist è in ascolto e avvia una connessione.

Analogamente, il processo Receptionist si mette in attesa di connessioni sulla porta mediante l'utilizzo di un oggetto della classe ServerSocket di Java, dichiarato all'interno della classe.

Una volta che il processo Receptionist è pronto ad accettare una richiesta, il cliente comunica il numero di posti nell'apposito campo dell'interfaccia che gli si presenta.

Ricevuta l'informazione, il processo Receptionist verifica se ci sono abbastanza posti liberi e in caso positivo assegna un tavolo al cliente, altrimenti gli comunica la mancata disponibilità e di riprovare in seguito.

Se il cliente ottiene i posti desiderati, chiude la connessione con il receptionist, che si mette in attesa di altre richieste, scannerizza il menù e inizia ad ordinare.

Altrimenti, il receptionist comunica al cliente il tempo di attesa necessario affinché si liberino dei posti.

A questo punto, il cliente può decidere di andarsene o di attendere.

Se il cliente decide di attendere, al termine del tempo di attesa gli verrà assegnato un tavolo.

Comunicazione cliente-cameriere (GetOrder)

Il cliente vuole comunicare con il cameriere per richiedere degli ordini.

Analogamente alla comunicazione con il receptionist, la classe Customer dichiara una socket e avvia una connessione con il processo Waiter, che nel frattempo ha dichiarato una server socket ed è in attesa di richieste su di una porta.

Una volta che il cameriere è pronto a prendere ordini, il cliente seleziona l'ordine desiderato dal menù, scannerizzato in precedenza e visualizzato sull'interfaccia.

Effettuato l'ordine, e atteso che gli sia arrivato, il suo conto viene aggiornato.

Una volta terminato di ordinare, il cliente chiude la connessione con il cameriere, che viene terminato, mediante il pulsante "Chiedi il Conto" dell'interfaccia e se ne va.

Comunicazione cameriere-cuoco

Il cameriere ha ricevuto un'ordinazione da un cliente e deve inviarla allo chef per prepararla.

Come descritto in precedenza, la classe Waiter dichiara una socket per avviare una comunicazione con il processo Chef, che intanto dichiara una server socket e si mette in attesa di ordini da preparare su di una porta.

Quando lo chef è pronto, il cameriere gli invia l'ordine da preparare e, una volta terminata l'attesa, riprende l'ordine e lo porta al cliente che lo ha ordinato, ripetendo il procedimento fintanto che il cliente vuole ordinare.

Accesso e Registrazione

Per avere la possibilità di utilizzare il sistema, il cliente dovrà prima completare una fase di login. Egli dovrà comunicare un username univoco ed una password per la registrazione negli appositi campi dell'interfaccia ed utilizzarli come credenziali di accesso per le successive volte.

Quando un cliente tenta di eseguire una registrazione, il sistema verifica se l'username dell'utente è disponibile e se l'utente non si è già registrato, eseguendo una query di ricerca all'interno della tabella Utenti del database.

Se l'username è già utilizzato, viene visualizzato un messaggio di errore.

Se invece l'utente completa la fase di registrazione, il sistema memorizza le sue credenziali all'interno della tabella Utenti del database mediante una query di inserimento e lo indirizza al form di login.

Quando il cliente tenta di eseguire il login, il sistema verifica se in precedenza ha effettuato la registrazione mediante le credenziali inserite, utilizzando una query di ricerca all'interno della tabella Utenti del database.

Gestione delle Password

Per garantire sicurezza, le password che i clienti utilizzano per l'accesso non saranno memorizzate in chiaro all'interno del database, nè saranno visibili al momento dell'inserimento nel campo dell'interfaccia.

Verrà infatti utilizzato un algoritmo di hashing come forma di crittografia.

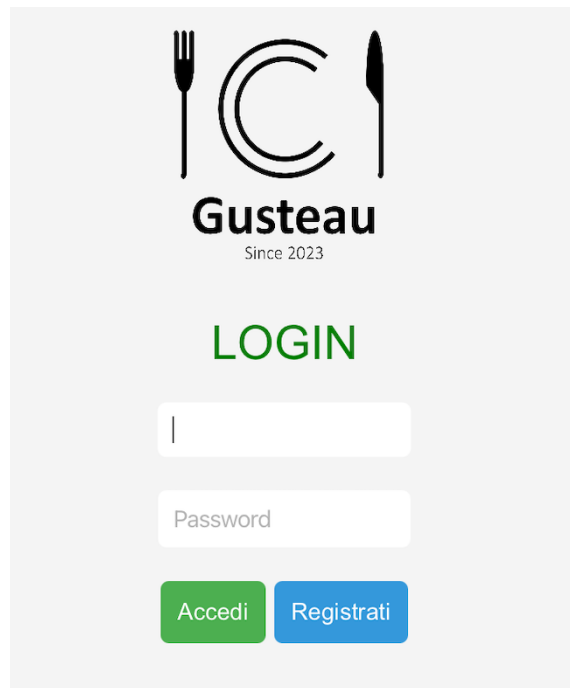
L'uso dell'algoritmo di hashing, che è irreversibile (non è possibile tornare alla stringa originale partendo solo da quella hashata), rende ipoteticamente impossibile a chi non possiede la password di potervi accedere.

Siccome un algoritmo di hashing è deterministico (a parità di stringa iniziale genera la stessa sequenza esadecimale), quando un cliente tenterà di eseguire la fase di login, il sistema, prima di verificare che le sue credenziali siano corrette ricercandole all'interno del database, utilizzerà lo stesso algoritmo di hashing applicato durante la fase di registrazione per convertire la password.

La mancata visibilità della password al momento dell'inserimento è garantita dall'elemento PasswordField di JavaFX.

Per garantire un'ulteriore sicurezza, gli utenti dovranno utilizzare password contenenti almeno 8 caratteri, tra cui almeno 1 carattere speciale, 1 lettera maiuscola e 1 numero.

Tale controllo è eseguito mediante l'utilizzo di una espressione regolare.

The image shows a login form for 'Gusteau'. At the top, there is a logo consisting of a fork, a plate, and a knife, with the text 'Gusteau' and 'Since 2023' below it. Below the logo, the word 'LOGIN' is written in green. There are two input fields: one for a username (with a vertical line as a placeholder) and one for a password (labeled 'Password'). At the bottom, there are two buttons: a green 'Accedi' button and a blue 'Registrati' button.

Form di Login

Rispettare lo standard aiuta a ridurre le probabilità che un utente malintenzionato si appropri della password di un altro utente e ad aumentare la difficoltà di un approccio brute-force.

Richiesta Posti

Una volta che il cliente ha effettuato l'accesso, deve comunicare al receptionist il numero di posti che gli servono.

Il sistema cattura il numero inserito dal cliente nell'apposito campo dell'interfaccia e lo invia al receptionist mediante una `println` sulla socket.

Siccome la `println` è bloccante, il cliente resterà in attesa di una risposta.

Il receptionist ha a disposizione il numero di posti e di tavoli disponibili del ristorante, ed usa queste informazioni per accettare o meno la richiesta del cliente.

Quando un cliente gli comunica un numero di posti, esso controlla se c'è un tavolo disponibile con i posti necessari e, in caso positivo, assegna aleatoriamente un tavolo al cliente, altrimenti comunica al cliente il tempo di attesa da rispettare affinché si liberino dei posti.

Per simulare il liberamento dei posti occupati, nel receptionist è definito uno scheduler che ad intervalli di tempo aleatori libera i tavoli.

Un altro scheduler è utilizzato per calcolare, sempre in maniera casuale, il tempo di attesa che deve rispettare il cliente affinché si liberino dei tavoli.

Per evitare tempi di attesa lunghi, l'unità di tempo utilizzata è il secondo.

Ordinazioni

Una volta preso posto e scannerizzato il menù, il cliente può ordinare tutto ciò che vuole cliccando sugli ordini del menù dall'interfaccia.

Il sistema cattura l'ordine selezionato e lo invia al cameriere.

Quando ha terminato le ordinazioni, il cliente può richiedere il conto mediante l'apposito pulsante.

La scelta dell'ordine dal menù permette di evitare di controllare che l'utente cerchi di ordinare un piatto non preparato nel ristorante.



Form Richiesta Posti



Esempio di Attesa

Scrittura Menu

Lo chef ha la possibilità di scrivere a proprio piacimento il menù e di stilare i prezzi degli ordini. Ogni volta che inserisce un ordine e il relativo prezzo negli appositi campi dell'interfaccia ad esso dedicata, il sistema controlla che tale ordine non sia già presente nel menù eseguendo una query di ricerca all'interno della tabella Ordini del database. In caso negativo, l'ordine e il suo prezzo sono inseriti nella tabella Ordini del database mediante una query di inserimento.

Durante la sua scrittura, lo chef avrà la possibilità di visualizzare in tempo reale il menù.

Naturalmente, avrà anche la possibilità di rimuovere ordini, semplicemente cliccando su uno di essi dal menù.

Analogamente all'inserimento, in questo caso verrà eseguita una query di cancellazione dalla tabella Ordini del database.

Sincronizzazione

Per evitare problemi di race condition, ogni thread ha una propria istanza personale delle risorse da utilizzare.

L'utilizzo di thread cameriere e chef dedicati ad ogni cliente, permette di evitare controlli ristretti di conflitti ed evita la possibilità di deadlock (attesa infinita del cliente) all'interno del sistema.

Tuttavia, siccome il processo receptionist è unico, bisogna necessariamente considerare il caso,

Esempio scrittura menù

assolutamente probabile, in cui più clienti in simultanea tentino di richiedere posti.

Se non venisse gestito correttamente, potrebbe verificarsi il caso paradossale in cui non ci sono più posti disponibili ma un cliente entra lo stesso, in quanto ha effettuato la richiesta prima dell'effettiva assegnazione dei posti.

Per far sì che ad ogni istante solo una richiesta venga gestita dal receptionist, è stato utilizzato un semaforo che garantisca l'accesso in mutua esclusione al receptionist.

Quando il receptionist accetta una richiesta di connessione da parte di un cliente, acquisisce il semaforo, effettua i controlli, fornisce la risposta al cliente e rilascia il semaforo.

In questo modo, se un cliente tenta di richiedere posti mentre lo sta facendo un altro, si metterà in attesa.

Questo permette di simulare l'effetto di una coda per avere accesso al receptionist.

Errori accidentali

Il sistema deve far fronte al caso, assolutamente plausibile, in cui un utente ordini erroneamente un piatto cliccando accidentalmente su di esso.

Per evitare che ciò avvenga, tutte le operazioni fornite dal sistema all'utente saranno controllate da una finestra di conferma.

Un ulteriore controllo effettuato, per verificare che l'utente inserisca correttamente le credenziali di accesso al momento della registrazione, è quello della conferma della password.

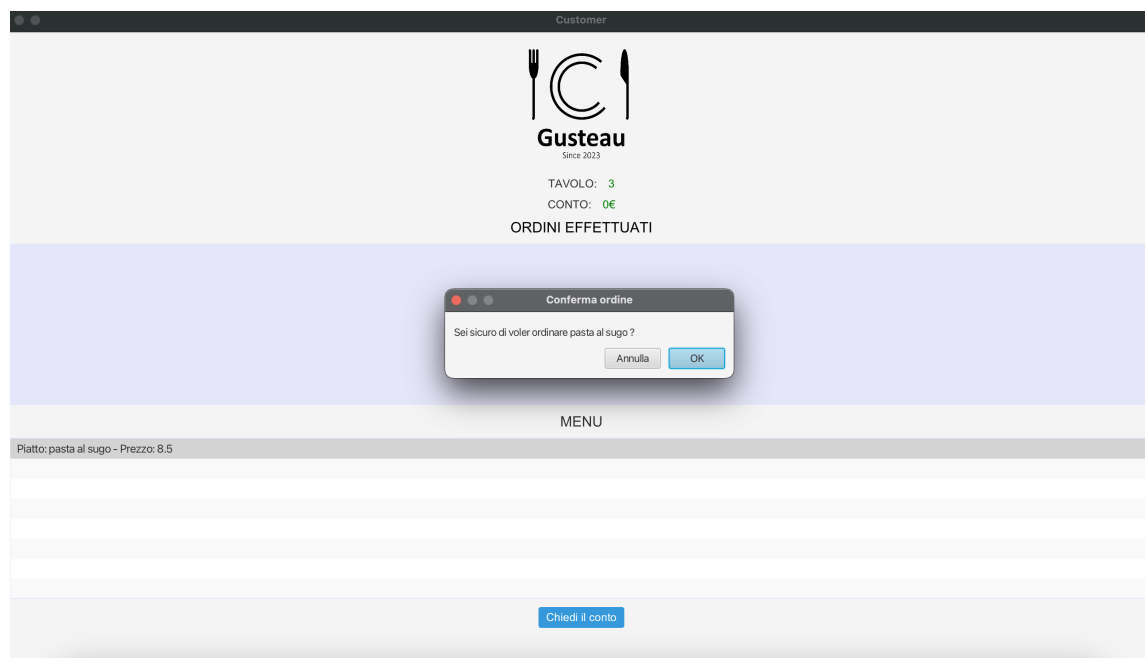
Al momento della registrazione, oltre ad inserire la password, il cliente dovrà confermarla in un altro campo.

Gestione delle interfacce

Per la gestione delle interfacce e il cambiamento di scena da una all'altra, sono stati dichiarati dei metodi che adattano dinamicamente la scena in funzione dello stato del sistema.

L'esecuzione del sistema inizia con la visualizzazione dell'interfaccia di Login, dalla quale si può giungere a quella di Registrazione, se si clicca sull'apposito pulsante, o a quella per la richiesta dei posti disponibili, se si esegue correttamente l'accesso.

Il meccanismo di cambio di scena, eseguito in seguito alla pressione dei pulsanti da parte dell'utente o all'occorrenza di eventi, prevede di selezionare la scena corrente catturando un suo elemento e di sostituirla caricando il contenuto del file FXML di un'altra interfaccia.



Esempio di errore di ordinazione

Manuale Utente e Guida all'Utilizzo

Il suddetto progetto è destinato alla realizzazione di un applicativo software.

L'idea è che i due attori principali, cliente e chef, debbano cliccare su di un'icona per eseguire l'applicazione, senza doversi interessare alla logica implementata per il funzionamento.

La loro interazione con il sistema deve essere limitata all'utilizzo delle interfacce.

All'utilizzatore del sistema, ad esempio il cliente, interessa solo effettuare ordini nel ristorante e pagare il conto, non il modo con il quale interagisce con i camerieri e il cuoco.

Creazione e Accesso al Database (MySQL per MacOS)

Per la gestione del database è necessaria l'installazione di un DBMS.

Come prima cosa va eseguito il server, cliccando sul pulsante "Start MySQL Server" e creato il database per la memorizzazione delle credenziali dell'utente e del menù in locale, cliccando sul pulsante "Initialize Database".

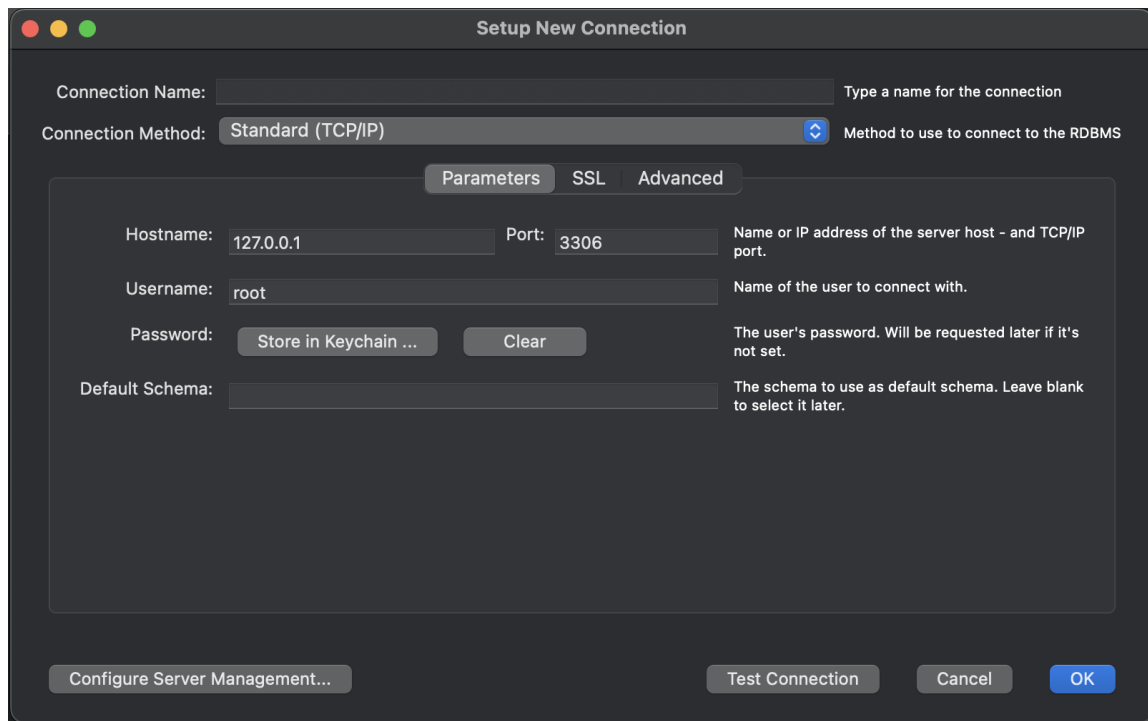
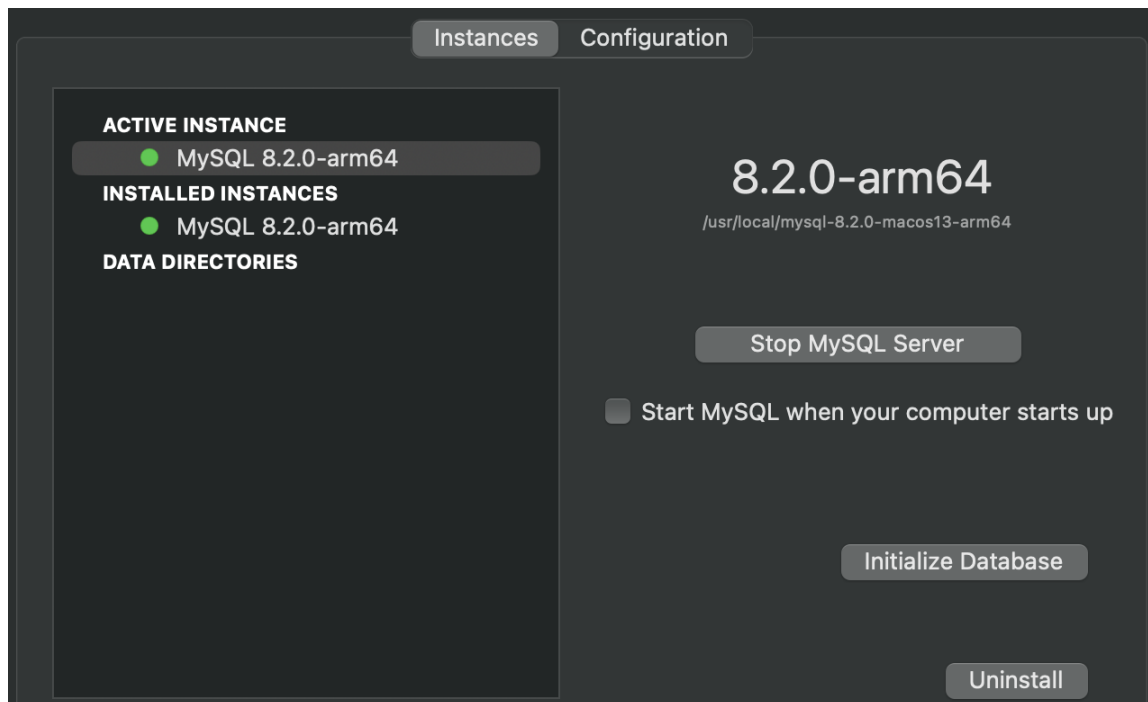
Nella schermata che comparirà, bisognerà stabilire una password per l'accesso al database.

Ora bisogna aprire MySQL Workbench e creare una connessione al server del database.

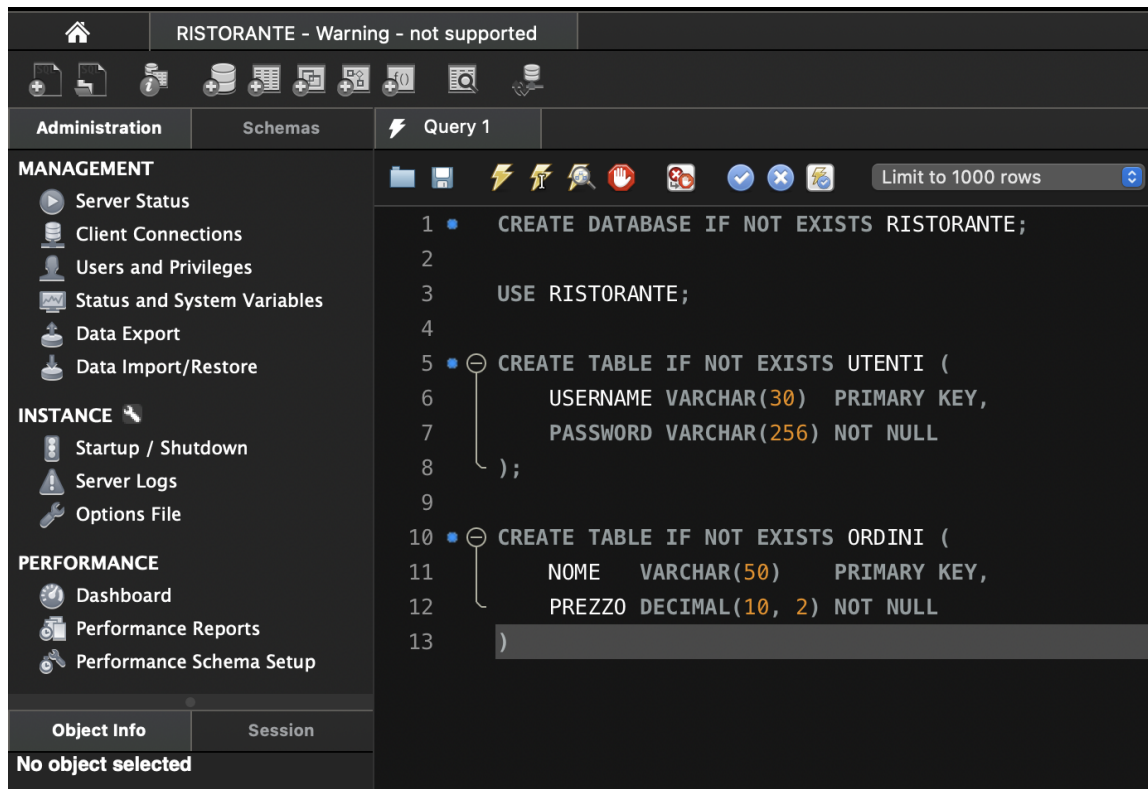
Nella seguente finestra, bisogna assegnare un nome arbitrario alla connessione (opportuno stabilire lo stesso nome del database), impostare il tipo di connessione (standard TCP/IP), inserire come Hostname l'indirizzo IP del dispositivo sul quale è eseguito il server (in questo caso "localhost" o 127.0.0.1 in quanto il server è in locale), stabilire la porta da utilizzare (di default la 3306) e l'username dell'utente con il quale si vuole eseguire l'accesso (lasciare "root" per indicare che si è l'amministratore del database).

Creata la connessione e inizializzato il database, è tempo di inizializzare le tabelle eseguendo il codice memorizzato nello script SQL.

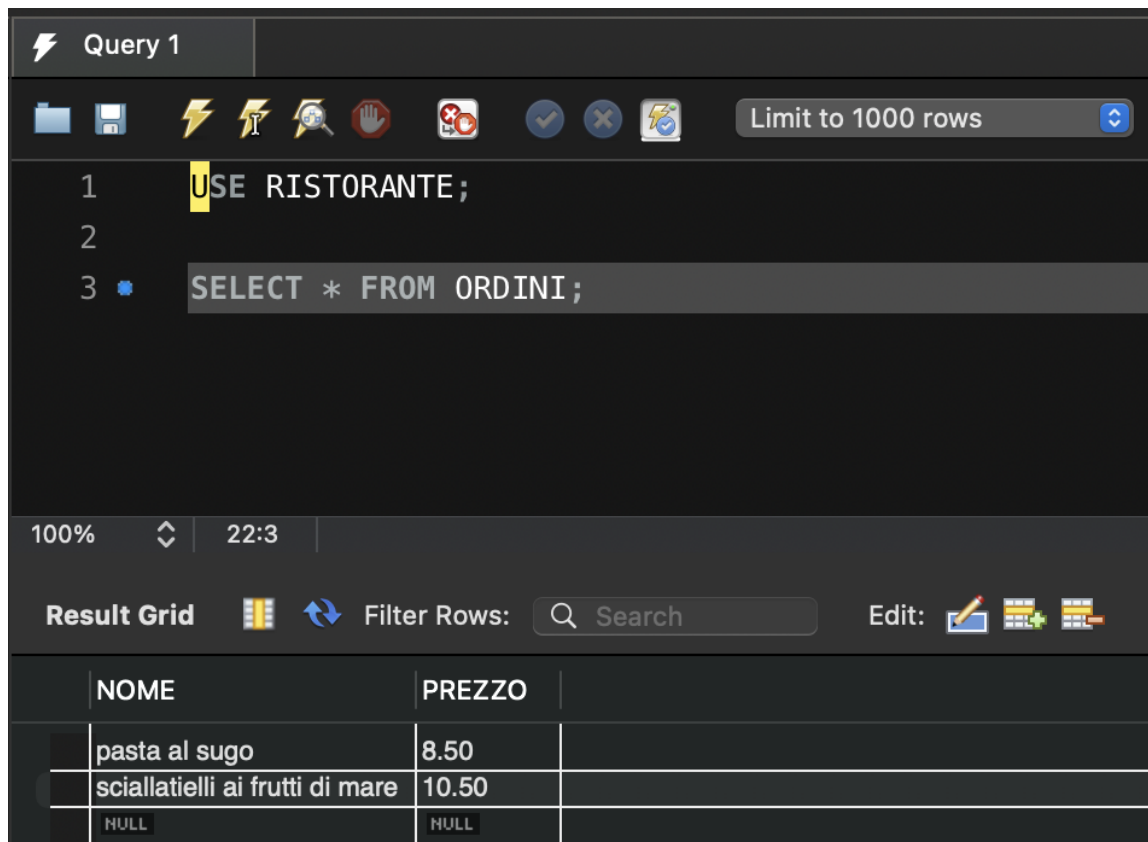
Ora il database è pronto e per verificarne il funzionamento si possono eseguire query di esempio.



Esempio di creazione della connessione



Script di creazione del database



Esempio di query

Compilazione ed Esecuzione (IntelliJ e Maven per MacOS)

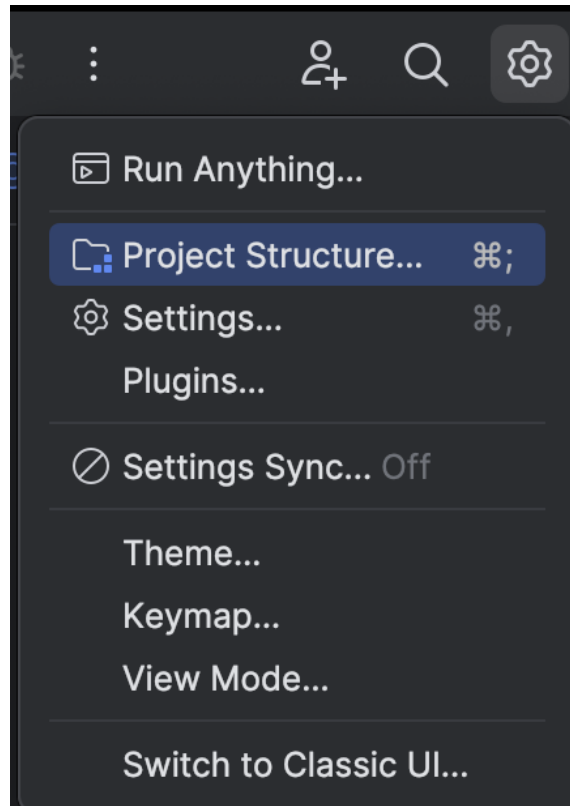
Il progetto necessita di alcuni plugin per essere compilato ed eseguito correttamente.

JavaFX

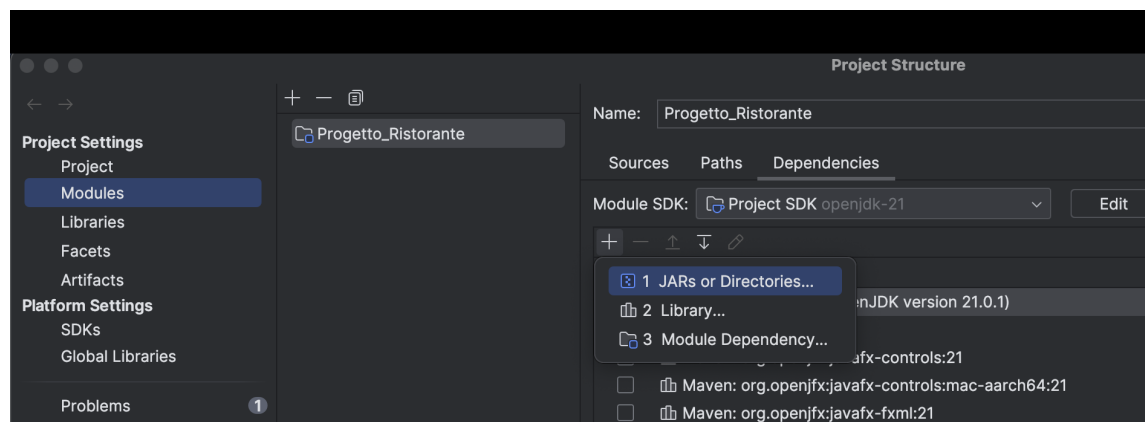
Per il funzionamento delle interfacce sono necessarie la libreria JavaFX.

Successivamente bisogna includere la libreria scaricata nella struttura del progetto.

Con IntelliJ aperto, aprire le Impostazioni in alto a destra e selezionare l'opzione "Project Structure" dal menù a tendina che compare.



Ora bisogna selezionare la sezione "Modules", cliccare sul simbolo "+", selezionare l'opzione "JARs or Directories" e selezionare il file JAR scaricato precedentemente per JavaFX.



Connector

Per garantire la connessione al database è necessario il plugin Connector.

La procedura per includere il plugin nel progetto è la medesima utilizzata per includere la libreria JavaFX.

Per verificare che la connessione al database e che l'inclusione della libreria JavaFX siano andate a buon fine, si può eseguire una tra le classi Customer.java e Chef.java ed utilizzare il software.