

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 consentirà ad un cliente di comunicare il numero di posti necessari e di effettuare varie ordinazioni, e allo chef di scrivere il menù a proprio piacimento.

I clienti, una volta terminata una fase di registrazione o accesso, avranno accesso al menù e potranno effettuare il numero di ordini che desiderano. Inoltre interagiranno, in maniera diretta, con un receptionist, che avrà il compito di verificare le disponibilità, per prendere posto, e, in maniera indiretta, con i camerieri per richiedere e ottenere ordini.

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.

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

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 cliente fungerà da client, in quanto dovrà prima comunicare al receptionist il numero di posti di cui necessita e poi al cameriere gli ordini desiderati
- Il receptionist fungerà da server, in quanto si metterà in attesa di richieste da parte di clienti
- Il cameriere 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
- Lo chef fungerà da server, in quanto si metterà in attesa di richieste di preparazione di ordini da parte dei camerieri.

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.

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 ed avranno delle classi Controller che si occuperanno del loro funzionamento.

Ogni interfaccia sarà gestita e stilizzata mediante un apposito file FXML.

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

Comunicazione

Ogni entità avrà una porta ed una socket dedicate per permetterle di comunicare con le altre. 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.

Comunicazione cliente-receptionist (GetRequiredSeats)

Il cliente vuole comunicare con il receptionist per richiedere dei posti. Dichiarare una socket utilizzando un oggetto della classe Socket di Java, specificando il suo indirizzo (localhost) e la porta del receptionist e avviare una connessione. Analogamente, il receptionist si mette in attesa di connessioni sulla porta mediante l'utilizzo di un oggetto della classe ServerSocket di Java. Una volta che il receptionist è pronto ad accettare una richiesta, il cliente comunica il numero di posti nell'apposito campo dell'interfaccia che gli si presenta. Ricevuta la richiesta, il 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.

Comunicazione cliente-cameriere (GetOrder)

Il cliente vuole comunicare con il cameriere per richiedere degli ordini. Analogamente alla comunicazione con il receptionist, il cliente dichiara una socket e avvia una connessione con il cameriere, che nel frattempo ha dichiarato una server socket ed è in attesa di richieste. 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 e può effettuare un nuovo ordine. 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.

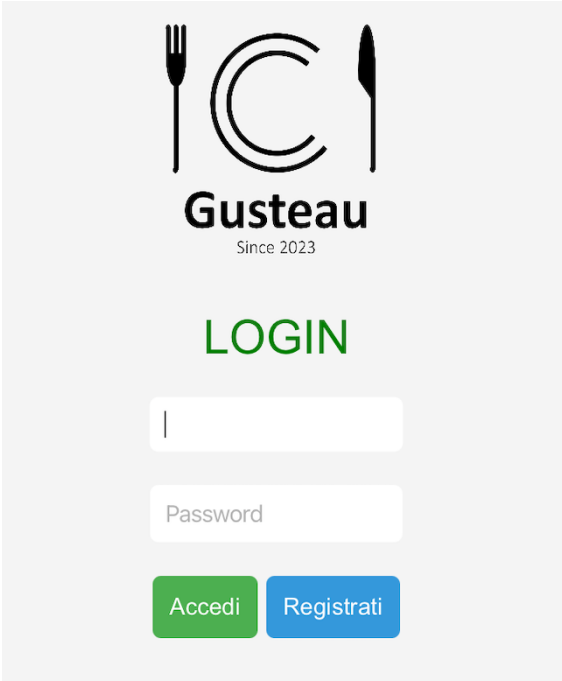
Comunicazione cameriere-cuoco

Il cameriere ha ricevuto un'ordinazione da un cliente e deve inviarla allo chef per prepararla. Come descritto in precedenza, il cameriere utilizza una socket per avviare una comunicazione con lo chef, che intanto utilizza una server socket e si mette in attesa di ordini da preparare sulla 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. Esso 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 una volta eseguendo una query di ricerca all'interno della tabella Utenti del database. Se l'username è già utilizzato, viene visualizzato un messaggio di errore, mentre se l'utente si è già registrato una volta, viene indirizzato all'interfaccia di login. 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. Quando il cliente tenta di eseguire il login, il sistema verifica se l'utente in precedenza ha effettuato

la registrazione mediante le credenziali inserite utilizzando una query di ricerca all'interno della tabella Utenti del database.

The image shows a login interface for a restaurant named 'Gusteau'. At the top, there is a logo consisting of a fork, a plate with the letter 'C', and a knife, with the text 'Gusteau' and 'Since 2023' below it. Below the logo, the word 'LOGIN' is displayed in green. There are two input fields: the first is empty with a cursor, and the second is labeled 'Password'. At the bottom, there are two buttons: a green one labeled 'Accedi' and a blue one labeled 'Registrati'.

Form di Login

Gestione delle Password

Per garantire sicurezza, le password che i clienti utilizzano per l'accesso non saranno memorizzate in chiaro all'interno del database.

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.

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. 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. 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.



Benvenuto! Di quanti posti hai bisogno?

Form Richiesta Posti



Non ci sono abbastanza posti disponibili, vuoi attendere 5 minuti ?

Esempio di Attesa

Ordinazioni

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

Il sistema cattura il testo dell'ordine selezionato, lo invia al cameriere ed esegue una query di ricerca nella tabella Ordini del database per cercare il prezzo dell'ordine e aggiungerlo al conto del cliente. Quando ha terminato le ordinazioni, il cliente può richiedere il conto mediante l'apposito pulsante. La scelta dell'ordine dal menù permette di evitare il controllo che l'ordine richiesto dal cliente sia effettivamente presente.

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.

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

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

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

Manuale Utente e Guida all'Utilizzo

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

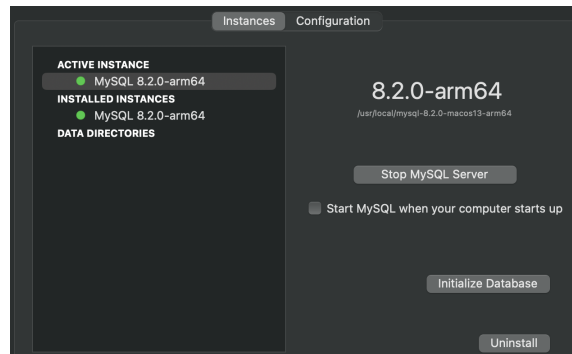
L'idea è che i due attori principali, cliente e chef, debbano cliccare su di un'icona per eseguire l'applicazione, senza doversi interessare alla logica implementata per il funzionamento.

Creazione e Accesso al Database (MySQL per MacOS)

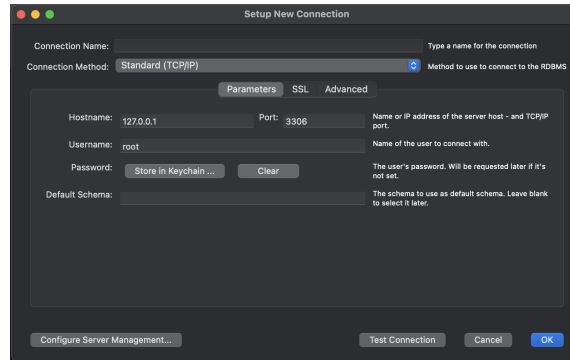
Per la gestione del database è necessaria l'installazione di un DBMS.

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".

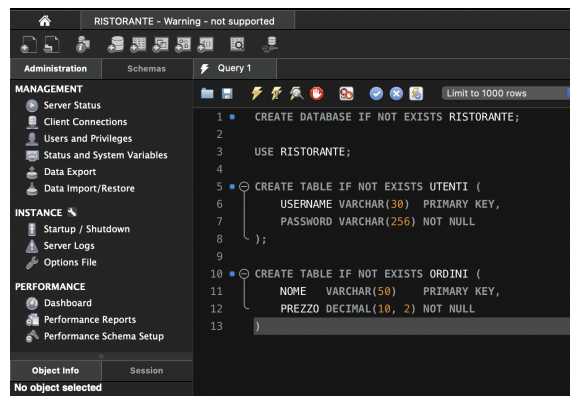
Nella schermata che comparirà, bisognerà stabilire una password per l'accesso al database.



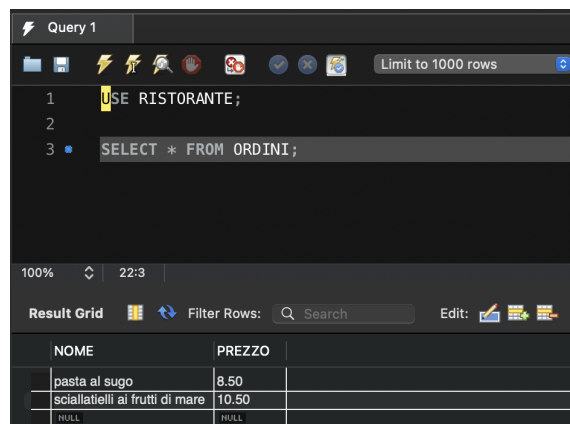
Ora bisogna aprire MySQL Workbench e creare una connessione al server del database. Nella seguente finestra, bisogna assegnare un nome arbitrario alla connessione (opportuno stabilire lo stesso nome del database), impostare il tipo di connessione (standard TCP/IP), inserire come Hostname l'indirizzo 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.



Compilazione ed Esecuzione (IntelliJ e Maven per MacOS)

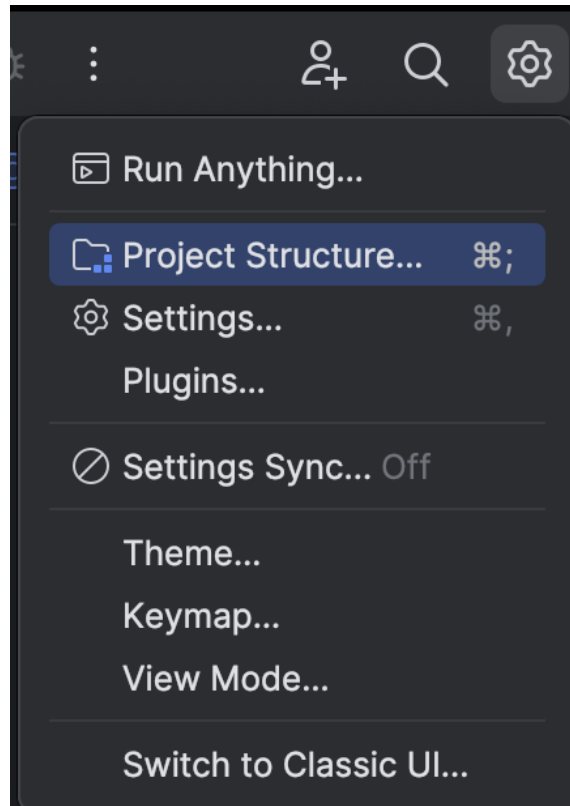
Il progetto necessita di alcuni plugin per essere compilato ed eseguito correttamente.

JavaFX

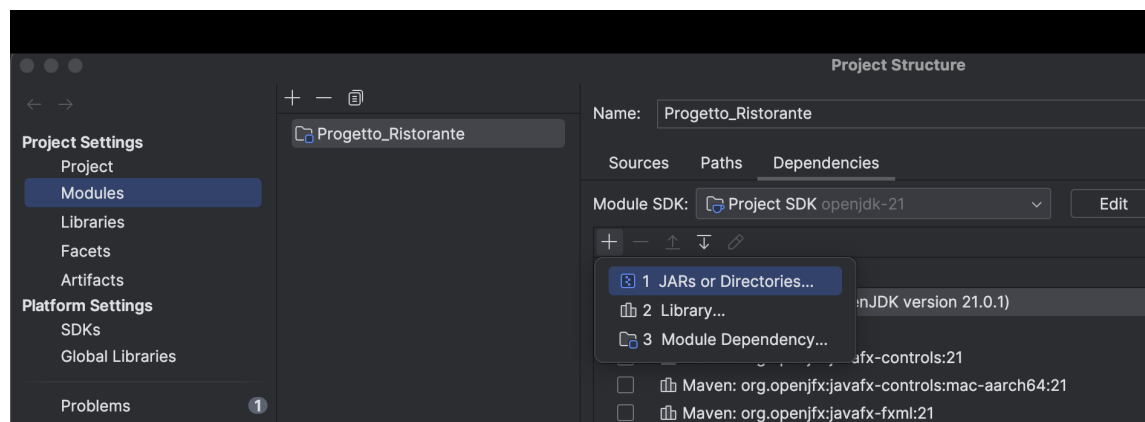
Per il funzionamento delle interfacce sono necessarie la libreria JavaFX.

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.



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.

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.