Tre pagine, una per il guest e la ricerca dei ristoranti, una area personale divisa in utente registrato e guest, una per le specifiche del ristorante(recensioni, prenotazioni ecc…)

Ristorante

Nome del ristorante

• Informazioni sul luogo in cui si trova il ristorante, in particolare: (Nazione • Città • Indirizzo • Latitudine Longitudine)

• Fascia di prezzo (prezzo medio in euro)

• Disponibilità del servizio di delivery (si/no)

• Disponibilità del servizio di prenotazione online (si/no)

• Tipo di cucina

OWNERID

Somma recensioni

Valutazione del singolo ristorante

Ristorante service

Visualizza i ristorant(luogo fascia oraria servizi e cerca)

Vedere recensioni e commenti NON scrivere

Registrarsi

Metodi:

cercaRistorante(string nome ristorante) : list<Ristorante>

aggiungiRistorante()

MainviewController

visualizzaRistorante(int idristorante) : redirect a pagina reataurantview

restaurantviewController

visualizzaRecensioni()

User

Campi: • Nome • Cognome • Username • Password (cifrata) • Data di nascita (facoltativa) • Luogo del domicilio – lista di ristoranti preferiti -gestione recensioni(ins, elim, mod, da 1 a 5 stelle) lista di recensioni

Service User

registrazione(User utenteRegistrante)

aggiungiRistorantePreferito()

rimuoviRistorantePreferito()

getPreferiti()

UserviewController

visualizzaPreferiti()

Owner(eredita da user)

lista di ristoranti -valutazione singolo ristorante -valutazione complessiva del ristoratore (media dei ristoranti) -numero recensioni singolo ristorante -numero recensioni del ristoratore (somma dei ristoranti)

OwnerService

PRENDI TUTTI I RISTORANTI CHE HANNO COME OWNER QUESTO, DAL REVIEWSERVICE CHIAMA IL “GETMEDIARECENSIONIRISTORANTE(RESTAURANTID)” E FAI LA MEDIA DI QUELLE

Foreach (ristorante in restaurantService.getAllRestaurantsByOwner(ownerID))

Somma+= reviewService.getAverageRating(ristorante.getID())

Count++

Somma/count;

OwnerviewController

aggiungiRistorante():richiama un metodo che richiama l’aggiungi ristorante di restaurant

visualizzaRiepilogo()(con media delle stelle e numero delle recensioni) locale

visualizzaRistoranti()

visualizzaRecensioni() con il dettaglio delle stelle associate ad ognuna

Il ristoratore è nello stesso json del dello user, i model sono due diversi, ma il service è uno solo. Il DAO è uno solo.

Rewiew

-stelle -Risposta(tipo messaggio)

Al posto della classe messaggio usiamo recensioni con stelle a null. ()

Review service

aggiungiRecensione()

modificaRecensione()

eliminaRecensione()

rispostaRecensione()

getAvarageRating()

Review Controller

inIt()

answer()

Manuale Utente

Alto livello funzionamento del programma screenshot

Come usare il programma come installare l’ambiente (es l’applicazione è stata testata su windows 11 con java 21.1)

Utilizzare un livello di astrazione e un linguaggio adeguati ad un pubblico non esperto e non tecnico

Manuale utente diviso in 4 sezioni

Come installare

Utente

Cliente

Ristoratore

Manuale tecnico

report per altri sviluppatori (giustificare le scelte fatte es. algoritmi o strutture dati complessità ec…)

scelte architetturali

strutture dati utilizzate

scelte algoritmiche

javadoc è parte del manuale

come sono fatti i file i cui sono memorizzati i dati (design pattern?)

possibili estensioni dell’applicazione (es con certi tipi di dati abbiamo problemi quindi una possibile aggiunta sarebbe …)

Design manuale

frontespizio titolo autori data versione documento

report tecnico della soluzione sviluppata (scelte di architettura)

limiti della soluzione sviluppata

sitografia/bibliografia

JAVADOC

/\*\* commento collegato alla classe o altro \*/

Vengono generate delle pagine HTML che descrivono le pagine della soluzione

Esempio

/\*\*

\* questo fjjdiij un commento

\* <strong>sono</strong>

\*/

Tag @name comment

Ogni tag devono essere su una nuova riga

Prima della definizione del metodo ci sarà il blocco di commento atto a spiegarne il funzionamento

/\*\*

\*@author niggabos

\*

\*/

Public class drama {…}

@param <nome parametro> fornire una breve descrizione sul parametro

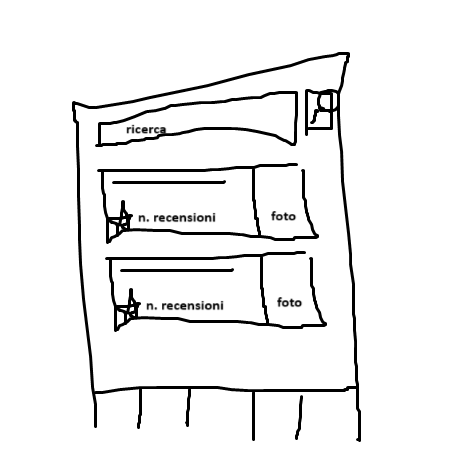
@return cosa restituisce il metodo

PACKAGE

Usare i tag @see per riferirsi alle classi metodi attributi in esso contenuti

GITHUB

clone



^ esempio di layout possibile

La recensione oltre ad avere tutti i suoi dati ha due ID uno del ristorante a cui è stata fatta la recensione e uno del cliente che ha fatto la recensione