Pro idee

### recensioni e risposte

pensando a come gestirle abbiamo pensato a vari modi, tra cui gestire come semplici campi stringhe appartenenti a recensioni, poi come classi separate e siamo giunti a una conclusione

Abbiamo deciso di gestire le recensioni con la classe recensione; per quanto riguarda le risposte alle recensioni utilizzeremo comunque la classe recensione ma con i campi non necessari per una risposta a null: un esempio è il campo valutazione, che verrà messo appunto a null nel caso delle risposte

### UTENTI

è registrato o non registrato

abbiamo deciso di usaree solo le classi utente registrato e ristoratore poichè la schermata verrà gestità dal gestorte e non sarà necessario salvare in formazioni in caso non fossi registrato

ristoratore estende utente registrato.

La classe verrà gestita dall’user service e user controller

### SERVICE E CONTROLLER

Per svolgere tutte le operazioni relative all’applicazione utilizzeremo delle classi service e controller che si occuperanno di questo lato del progetto, ad esempio: ad esempio per la classe user avremo user service che avrà metodi come registra utente, e user controller avrà visualizza ristoranti preferiti (si occuperà della parte grafica della pagina)

Gestione file

Nonostante la lentezza riscontrata operando sui file json abbiamo deciso di continuare a utilizzare questo formato poiché risulta essere l’unico disponibile. Utilizzare un db renderebbe tutto più veloce e semplice, ma purtroppo non abbiamo la possibilità di sfruttarlo.

Pagine dell’applicazione

L’applicazione avrà 3 principali pagine, che varieranno in base a che tipo di utente risulta loggato: la pagina di ricerca e visualizzazione dei ristoranti, la pagina del singolo ristorante con informazioni e recensioni, e la pagina per l’area riservata dell’utente.