Progetto d’esame

Tecnologie e applicazioni web, a.a 2018/2019

Si realizzi un’applicazione web, comprensiva di server con API stile REST e front-end di tipo SPA, che permetta la gestione delle ordinazioni di un ipotetico ristorante/pizzeria. Il sistema deve prevedere le seguenti tipologie di utenti:

Camerieri

Ciascun cameriere porta con sé un dispositivo mobile (smartphone o tablet) con cui può prendere ordinazioni ai tavoli.

Cuochi

I cuochi o pizzaioli vengono notificati in tempo reale sul cibo ordinato da ciascun tavolo mediante i camerieri. Gli ordini sono inseriti in una coda e raggruppati per tavolo e per tempo di realizzazione di modo tale che la preparazione dei piatti per uno stesso tavolo possano essere serviti contemporaneamente. I cuochi, attraverso la loro interfaccia, possono notificare il sistema dell’inizio e la fine di una preparazione. Quando le preparazioni di un determinato tavolo sono ultimate, il cameriere associato a quel tavolo viene notificato di tale evento e può quindi procedere al servizio.

Baristi

I baristi sono concettualmente simili ai cuochi ma si occupano solo delle bibite. I baristi ricevono notifica delle ordinazioni di ciascun tavolo e inviano notifica di fine preparazione ai camerieri quando il “vassoio” di bibite di un determinato tavolo è pronto per il servizio

Cassa

L’addetto alla cassa può “chiudere il conto” di un tavolo producendo uno scontrino con il totale da pagare. La cassa ha inoltre visione delle statistiche sullo stato degli ordini, in particolare:

- Quali tavoli hanno ordini in fase di preparazione
- Quale cameriere è associato a ciascun tavolo
- Quali tavoli sono liberi/occupati
- Quanto è lunga la coda di preparazioni della cucina

Infine, la cassa può conoscere il guadagno totale di una giornata.

Architettura

Il sistema deve essere composto da:

- Un web service di **backend** con API in stile REST, implementato in Javascript o Typescript su ambiente Node.js. Il servizio deve inoltre utilizzare:
    - Il DBMS MongoDB per gestire la persistenza dei dati
    - Il middleware Express.js per la gestione del routing

- Una web application di **front-end** in stile SPA realizzata con il framework Angular. L'applicazione, oltre alla versione web standard, comprende anche:
    - Un'applicazione mobile ibrida realizzata con Apache Cordova
    - Un client desktop realizzato con Electron

Le applicazioni mobile e desktop sono sostanzialmente identiche alla web application Angular browser-based ma possono includere funzionalità aggiuntive, a discrezione dello studente (ad esempio l'applicazione mobile può presentare elementi di interfaccia tipici del contesto mobile). L'applicazione mobile può utilizzare componenti e funzionalità fornite dal framework Ionic.
Il front-end realizzato è unico per ciascuna tipologia di utente precedentemente descritta. A seconda del ruolo dell'utente a seguito del login verranno presentate pagine e funzionalità diverse.

Funzionalità

Il sistema deve implementare le seguenti funzionalità:

1. Gestione degli utenti
    a. Registrazione di nuovi utenti con i seguenti ruoli: camerieri, cuochi, baristi e cassa. Il ruolo "cassa" gode del ruolo di amministratore;
    b. Possibilità di cancellazione di utenti esistenti da parte degli amministratori;
    c. Login degli utenti con le proprie credenziali di accesso. **Tutte le funzionalità del sistema sono accessibili previo login obbligatorio**.
2. Gestione dei tavoli e ordini (solo per camerieri)
    a. Modifica dello stato di un tavolo da *libero* a *occupato* da un certo numero di clienti. Il numero di clienti deve essere compatibile con i posti a sedere al tavolo;
    b. Aggiunta di una o più ordinazioni di piatti per un determinato tavolo. Quando le ordinazioni sono ultimate queste vengono notificate ai cuochi;
    c. Aggiunta di una o più ordinazioni di bevande per un determinato tavolo. Quando le ordinazioni sono ultimate queste vengono notificate ai camerieri;
    d. Notifica al cameriere addetto ad un tavolo dei piatti e bevande pronte da servire per quel tavolo.
3. Gestione della coda di preparazione dei cibi (solo cuochi):

    a. Visualizzazione in tempo reale della coda di preparazione dei cibi per ciascun tavolo. Si assuma per semplicità che vi sia un'unica coda di preparazione e che ciascun tavolo venga servito con logica FIFO. I cibi da realizzare vanno ordinati per tempo di preparazione, dal più lungo al più breve. Il cuoco, per ciascun cibo, può notificare al sistema l'inizio e la fine della preparazione. Quando tutti i cibi di un determinato tavolo sono stati preparati, il cameriere addetto a quel tavolo viene notificato che può procedere al servizio.
4. Gestione della coda di preparazione delle bevande (solo baristi):
    a. Visualizzazione in tempo reale della coda di preparazione bevande per ciascun tavolo. Quando il vassoio delle bevande è pronto il barista notifica tale evento al sistema che provvederà ad inoltrarlo al cameriere addetto a quel determinato tavolo (il funzionamento è analogo a quanto avviene per i cuochi).
5. Gestione cassa (solo per cassieri):
    a. Calcolo del conto totale di un tavolo e creazione dello scontrino con la lista delle ordinazioni consumate. Il tavolo passa quindi dallo stato *occupato* allo stato *libero* e può essere riutilizzato dai camerieri;
    b. Visualizzazione dei tavoli liberi e occupati;
    c. Visualizzazione degli ordini effettuati e in attesa di preparazione per ciascun tavolo;
    d. Visualizzazione statistiche su ciascun cameriere e cuoco (numero clienti serviti, piatti realizzati, etc.).

I gruppi sono liberi di implementare funzionalità aggiuntive oltre quelle precedentemente elencate. L'implementazione o meno di funzionalità aggiuntive non preclude né garantisce il conseguimento della lode ma contribuisce a definire un giudizio positivo.

Il sistema deve, in fase di avvio, precaricare nel database con una lista (anche ristretta) di cibi, bevande e tavoli per permettere il test del sistema a seguito della creazione degli utenti.

Consegna

La consegna avviene esclusivamente mediante piattaforma moodle alla pagina:

https://moodle.unive.it/mod/assign/view.php?id=101991

Ciascun gruppo deve consegnare un solo file, in formato zip, chiamato `<nomegruppo>.zip`.

Il file zip al suo interno deve contenere:

- La relazione di ciascuno studente, in formato pdf col nome `<nome>_<cognome>_<matricola>.pdf`
- Un file README.txt che spiega chiaramente come eseguire l'intera applicazione. Ad esempio, la lista di comandi da dare per installare le dipendenze (npm install o altro), eseguire il processo del DBMS, il server e i relativi client;

- Tutti i sorgenti necessari alla sua esecuzione.

NOTA: Non consegnare le librerie esterne installate attraverso il gestore pacchetti. In altre parole, tutte le directory `node_modules` devono essere vuote.

Relazione

Congiuntamente al progetto, ciascuno studente deve consegnare una relazione.
La relazione va compilata in forma strettamente individuale e porrà le basi per la discussione orale del progetto.

La relazione deve contenere:

- Una descrizione dell'architettura del sistema, quali sono i componenti e in che modo questi concorrono a soddisfare le features richieste;
- Una descrizione del modello dei dati utilizzato. Quali sono le collezioni e qual'è la struttura dei documenti di ciascuna collezione che vengono memorizzati nel database;
- Una descrizione delle API fornite dalla componente server. La descrizione deve contenere in modo chiaro la lista degli endpoints, degli eventuali parametri e il formato dei dati (JSON) che vengono scambiati nelle richieste HTTP;
- Una descrizione di come è stata realizzata l'autenticazione degli utenti, con relativo workflow;
- Una descrizione del client web realizzato con il framework Angular. In particolare, la lista dei vari Components, Services e delle Routes;
- Alcuni esempi, corredati possibilmente da screenshots, del workflow tipico dell'applicazione.

Altre informazioni

Per domande o chiarimenti è sempre possibile contattare il professore via mail all'indirizzo filippo.bergamasco@unive.it

Per domande sulla modalità di esame si faccia inoltre riferimento alla seguente pagina:

https://moodle.unive.it/mod/page/view.php?id=88040

Filippo Bergamasco,
Tecnologie e Applicazioni Web, a.a. 2018/2019