TAWRESTaurant

progetto di Tecnologie e Applicazioni Web

Questo progetto offre un sistema di gestione di uno pseudo-ristorante.

Introduzione:

Il back-end consiste in un server API scritto in TypeScript, che viene eseguito in ambiente node.js, e un database mongoDB per la persistenza dei dati.

Il front-end è basato su Angular ed offre 3 interfacce:

- Web, accessibile ovunque e in stile minimale, vanta una compatibilità con numerosi dispositivi grazie ad un design responsivo che si adatta a schermi di diverse dimensioni.
- Mobile, WebApp creata con cordova, come il sito web, ma in formato app. Più adatta per dispositivi mobili.
- Desktop, app multipiattaforma basata su Electron, permette di usare l'applicazione come se fosse un programma nativo del computer.

Oltre alla comunicazione tra backend e front-end tramite le API il server offre un elemento in tempo reale grazie a socket.io, questo permette un aggiornamento in tempo reale dei dati visualizzati per tutte le piattaforme.

Funzionalità ed implementazione:

1. **Gestione degli utenti:**
    a. I cassieri dalla pagina di gestione utenti possono registrare nuovi utenti premendo il bottone apposito. Dopo aver premuto il bottone si aprirà una finestra con una form per l'inserimento dei dati e del ruolo del nuovo utente. Lato server dopo aver verificato la validità dell'operazione i dati degli utenti vengono elaborati dal server per poi essere inseriti nel database.
    b. Nella gestione utenti è possibile eliminare singoli utenti premendo il bottone di fianco a ciascun utente e selezionando l'azione apposita. Lato server dopo aver verificato la validità dell'operazione verrà eliminato l'utente dal database.
    c. La pagina di login permette a tutti gli utenti registrati di inserire il proprio username e password per poi ricevere il token di autenticazione usato per accedere alle altre pagine. All'accesso l'username e la password vengono confrontati con i dati presenti nel database, prima si cerca l'utente con l'username e poi viene confrontata la password con l'hash memorizzato nel database.
2. **Gestione dei tavoli e degli ordini:**
    a. Il cameriere può occupare un tavolo dalla schermata dei tavoli. Qui c'è il bottone "occupa tavolo" che mostra una finestra di ricerca dove poter cercare i tavoli in base al numero di clienti da servire. Una volta scelto il tavolo tra i tavoli compatibili esso verrà occupato ed inserito nella sezione dei tavoli serviti. Il server verifica la compatibilità tra posti a sedere e numero di clienti prima di occupare il tavolo, una volta occupato il tavolo verrà aggiornato lo stato del tavolo nel database. Verrà lanciato un evento in socket.io per l'aggiornamento in tempo reale.
    b. Sui tavoli occupati si possono aggiungere degli ordini premendo l'apposito pulsante, si aprirà una finestra in cui si potranno aggiungere o rimuovere gli alimenti e bevande dagli ordini del tavolo. Dopo aver preso gli ordini il cameriere potrà notificarli

alla cucina e al bar premendo il pulsante apposito. Ogni ordine aggiunto, che sia alimento o bevanda, viene immediatamente memorizzato nel database per poi essere ripreso in un secondo momento. Dopo aver completato l'ordine il cameriere invia al database il comando per inoltrare gli ordini ai cuochi e baristi. socket.io gestisce l'aggiornamento in tempo reale con un evento.

c. Svolto assieme al punto **b**.

d. Una volta che tutte le bevande o tutti gli alimenti per un determinato tavolo sono pronti il cameriere riceverà un aggiornamento sul suo dispositivo. Compariranno 2 pulsanti a seconda degli ordini ultimati, se le bevande saranno pronte allora lui potrà segnalare l'avvenuta consegna con "bevande serite", nel caso degli alimenti potrà segnalare la consegna con "piatti serviti" L'aggiornamento in tempo reale è gestito dagli eventi di socket.io, dopo un evento il tavolo avrà i dati aggiornati. Per segnalare la consegna di alimenti o bevande il cameriere cambia lo stato del tavolo e lancia un evento su socket.io.

3. **Gestione della coda di preparazione dei cibi (solo cuochi):**
   a. La pagina di preparazione dei cibi per i cuochi è composta da una coda FIFO dei tavoli in base all'orario dell'ordine. Gli alimenti da preparare sono a loro volta elencati per ciascun tavolo ed ordinati per tempo di preparazione. Solo il primo piatto del primo tavolo potrà essere messo in preparazione. Ad ogni ordine messo in preparazione o notificato viene lanciato un evento per la sincronia di tutti i cuochi. Una volta che l'ultimo ordine di alimenti di un tavolo viene notificato un evento socket.io viene lanciato per notificare il cameriere.
4. **Gestione della coda di preparazione delle bevande (solo baristi):**
   a. Come per i cuochi i baristi vedranno un elenco dei tavoli in ordine FIFO in base all'orario dell'ordine e per ogni tavolo un

elenco di bevande da preparare, il funzionamento è uguale alla sezione della cucina.

5. **Gestione cassa (solo per cassieri):**
    a. Nella schermata di gestione dei tavoli i tavoli che hanno ricevuto sia gli alimenti che le bevande avranno un pulsate accanto per il calcolo del conto. Il conto verrà visualizzato in una finestra con un elenco dettagliato degli ordini. Dopo l'avvenuto pagamento il cassiere potrà liberare il tavolo chiudendo il conto con il pulsante. Dopo aver chiuso il conto verrà lanciato un evento per segnalare la disponibilità del tavolo nell'elenco dei tavoli occupabili.
    b. I tavoli vengono visualizzati nella pagina dei tavoli, tutti i tavoli vengono visualizzati in base allo stato. I tavoli liberi compariranno come liberi, invece quelli occupati avranno ulteriori informazioni come lo stato generale degli ordini, in preparazione, preparati o serviti. Per l'aggiornamento in tempo reale vengono usati eventi di socket.io.
    c. Gli ordini vengono elencati all'interno dei rispettivi tavoli nella pagina dei tavoli. Gli ordini avranno i dettagli come il nome e lo stato di preparazione. Per l'aggiornamento in tempo reale vengono usati eventi di socket.io.
    d. Le statistiche vengono visualizzate nella pagina degli utenti. Per i cuochi e baristi i piatti e bevande vengono assegnati alla fine della preparazione, invece il cameriere avrà il numero di clienti serviti aggiornato una volta che il tavolo avrà pagato il conto. L'aggiornamento delle statistiche è gestito internamente al server con le chiamate di notifica ordine e liberazione del tavolo.

Il backend:

è un server API stile REST implementato con node.js, express.js, mongoose.js e socket.io.

Node.js è l'interprete che esegue codice JavaScript

Express è un framework che semplifica l'implementazione degli endpoint di un server, permette di integrare middleware ed endpoint in modo veloce ed efficace.

Mongoose.js è una libreria che permette di interfacciarsi a mongoDB mettendo a disposizione funzioni di creazione e validazione di Collection e Document.

MongoDB è un database non relazionale con una struttura diversa da un database relazionale (SQL), infatti ha Document invece di Record, Collection invece di Tabelle.

Socket.io è una libreria che permette una comunicazione in tempo reale tra server e client.

La comunicazione è di tipo bidirezionale e basata su eventi.

MongoDB, la struttura dati.

Il database contiene 4 Collection:

1. **users**, contiene i dati per le 4 categorie di utenti:

   Campi generici appartenenti a tutte le categorie:

   _id: id alfanumerico automatico di mongoDB.

   username: nome utente univoco per accedere al server.

   name: nome utente.

   surname: cognome utente.

   role: ruolo dell'utente (cashier, cook, barman, waiter).

   salt: "sale" usato durante la cifratura della password.

   digest: risultato dell'hashing della password + il "sale".

   Cashier: nessun campo extra.

   Cook:

   totalPreparedDishes: numero di piatti preparati.

   Barman:

   totalPreparedBeverages: numero di bevande preparate.

   Waiter:

   totalServedCustomers: numero di clienti serviti

2. **menuitems**, contiene i dati degli alimenti e bevande:

   _id: id alfanumerico automatico di mongoDB.

   name: nome dell'alimento/bevanda.

   price: prezzo.

   preparationTime: tempo di preparazione.

   kind: Tipologia: alimento ("Food") o bevanda ("Beverage").

3. **orders**, contiene i dati delle 2 categorie di ordini:
   Campi generici appartenenti a tutte le categorie:
      _id: id alfanumerico automatico di mongoDB.
      table: _id del tavolo a cui appartiene l'ordine.
      status: stato dell'ordine ("pending", "preparing", "ready").
      kind: Tipologia: "FoodOrder" o "BeverageOrder".
   FoodOrder:
      food: _id dell'alimento ordinato.
      cook: _id del cuoco incaricato della preparazione.
   BeverageOrder:
      beverage: _id della bevanda ordinata.
      barman: _id del barista incaricato della preparazione.
4. **tables**, continene i dati dei tavoli:
   Table:
      _id: id alfanumerico automatico di mongoDB.
      number: numero univoco del tavolo.
      seats: numero di posti a sedere.
      status: stato del tavolo ("free", "not-served", "waiting", "served".
      numOfCustomers: numero di clienti seduti al tavolo.
      servedBy: Cameriere incaricato di servire il tavolo.
      occupiedAt: Orario in cui il tavolo è stato occupato.
      ordersTakenAt: Orario in cui il cameriere ha preso gli ordini.
      foodOrdersStatus: Stato degli ordini degli alimenti ("pending", "ready", "served").
      beverageOrdersStatus: Stato degli ordini delle bevande ("pending", "ready", "served").

Endpoints del server API a partire dall'endpoint "/api/v1":

endpoint	descrizione	request	response
GET /	restituisce un elenco degli endpoint disponibili.		oggetto JSON {api}
POST /login	restituisce un token dopo un'autenticazione con username e password.		oggetto JSON: {"token": JWT}
GET /menu	restituisce un array di menuitems in formato JSON.	query: name, price, preparationTime, kind	oggetto JSON [{menuitem}, {}, …]
POST /menu	inserisce nel database un nuovo elemento menuitem e lo restituisce in formato JSON.	body: {name, price, preparationTime, kind}	oggetto JSON {menuitem}
GET /menu/foods	restituisce un array di alimenti in formato JSON.	query: name, price, preparationTime, kind	oggetto JSON [{menuitem}, {}, …]
GET /menu/beverages	restituisce un array di bevande in formato JSON.	query: name, price, preparationTime, kind	oggetto JSON [{menuitem}, {}, …]
GET /menu/byId/:idM	restituisce il menuitem con l'id specificato.		oggetto JSON {menuitem}
PUT /menu/byId/:idM	modifica il menuitem con l'id specificato	body: {name, preparationTime, price}	
DELETE /menu/byId/:idM	elimina il menuitem specificato.		
GET /tables	restituisce un array di tables in formato JSON.	query: seats, status, servedBy, foodOrderStatus, beverageorderStatus	oggetto JSON [{table}, {}, …]

POST /tables	inserisce nel database un nuovo elemento table e lo restituisce in formato JSON.	body: {number, seats}	oggetto JSON {table}
GET /tables/byId/:idT	restituisce il table con l'id specificato.		oggetto JSON {table}
PUT /tables/byId/:idT	modificalo stato del table con l'id specificato.	query: action ("free", "occupy"),	
DELETE /tables/byId/:idT	elimina il table con l'id specificato.		
GET /tables/byId/:idT/orders	restituisce un array di ordini appartenenti al tavolo specificato.		oggetto JSON [{order}, {}, …]
PUT /tables/byId/:idT/orders	inoltra gli ordini in cucina, al tempo stesso cambia lo stato del tavolo e degli ordini.	query: action ("commit")	
GET /tables/byId/:idT /orders/:idO	restituisce l'order con l'idO specificato appartenente al tavolo con l'idT specificato		oggetto JSON {order}
PUT /tables/byId/:idT /orders/:idO	cambia lo stato dell'ordire e lo restituisce.	query: action ("assign", "ready")	oggetto JSON {order}
DELETE /tables/byId/:idT /orders/:idO	elimina l'ordine specificato.		
GET /tables/byId/:idT /orders/foodOrders	restituisce un array di ordini di alimenti del tavolo.	query: status	oggetto JSON [{order}, {}, …]
POST /tables/byId/:idT /orders/foodOrders	inserisce un ordine di alimenti nell'elenco degli ordini del tavolo.	body: {food}	oggetto JSON {order}
PUT /tables/byId/:idT /orders/foodOrders	cambia lo stato degli ordini di alimenti in "servito al tavolo"	query: action ("serve")	
GET /tables/byId/:idT /orders/beverageOrders	restituisce un array di ordini di bevande del tavolo.	query: status	oggetto JSON [{order}, {}, …]
POST /tables/byId/:idT /orders/beverageOrders	inserisce un ordine di bevande nell'elenco degli ordini del tavolo.	body: {beverage}	oggetto JSON {order}

PUT /tables/byId/:idT /orders/beverageOrders	cambia lo stato degli ordini di bevande in “servito al tavolo”	query: action (“serve”)	
GET /users	restituisce un array di utenti.		oggetto JSON [{user}, {}, …]
GET /users/cooks	restituisce un array di cuochi.		oggetto JSON [{user}, {}, …]
GET /users/barmans	restituisce un array di barman.		oggetto JSON [{user}, {}, …]
GET /users/cashiers	restituisce un array di cassieri.		oggetto JSON [{user}, {}, …]
GET /users/waiters	restituisce un array di camerieri.		oggetto JSON [{user}, {}, …]
POST /users/cooks	inserisce un cuoco nel database.	body: {username, name, surname, password}	oggetto JSON {user}
POST/users/barmans	inserisce un barman nel database.	body: {username, name, surname, password}	oggetto JSON {user}
POST /users/cashiers	inserisce un cassiere nel database.	body: {username, name, surname, password}	oggetto JSON {user}
POST /users/waiters	inserisce un cameriere nel database.	body: {username, name, surname, password}	oggetto JSON {user}
PUT /users/byId/:idU	cambia la password dell’utente con l’idU specificato.	body: {password}	
GET /users/cooks /byId/:idU/orders	restituisce un array di ordini assegnati al cuoco.		oggetto JSON [{order}, {}, …]
GET /users/barmans /byId/:idU/orders	restituisce un array di ordini assegnati al barman.		oggetto JSON [{order}, {}, …]
GET/users/waiters /byId/:idU/tables	restituisce un array di tavoli assegnati al cameriere.		oggetto JSON [{table}, {}, …]

Autenticazione:

La prima autenticazione da parte di un qualsiasi utente parte dalla pagina di login.

In questa pagina l'utente inserisce username e password.

La richiesta viene elaborata lato server.

Viene aggiunto un salt alla password e successivamente viene generato un hash che verrà confrontato con il digest salvato nel database.

Una volta verificata la password verrà inviato al client un token (JWT), il quale contiene le informazioni base dell'utente.

Dopo la prima autenticazione l'utente si autenticherà esclusivamente attraverso il token.

Tutti gli endpoint, esclusi api/v1 e login, sono protetti dall'autenticazione tramite token.

Alcuni endpoint sono specifici per alcuni utenti:

L'assegnamento dei tavoli ai clienti è compito del cameriere, così come la creazione e l'invio di ordini in cucina e la notifica dell'avvenuta consegna di essi al tavolo.

Invece la preparazione e notifica degli ordini è riservata a cuochi e barman.

Il cassiere può liberare un tavolo dopo aver generato il conto, inoltre ha poteri di amministratore, quindi può modificare dati sensibili come menu, tavoli e gestire il cambio password per tutti gli utenti.

L’applicazione Angular

è una SPA che permette agli utenti di autenticarsi dalla pagina di login.

In base al ruolo dell’utente vengono messe a disposizione diverse pagine.

Il cassiere può visualizzare lo stato dei tavoli e dei relativi ordini, può chiudere un conto, e gestisce i dati degli utenti.

Il cuoco e il barman possono visualizzare la coda di preparazione degli ordini, assegnare un ordine e notificare l’avvenuta preparazione.

Il cameriere può occupare i tavoli, controllare lo stato degli ordini dei tavoli a lui affidati e notificare quando sono stati serviti gli ordini.

I component sono:

- Navbar, barra di navigazione con informazioni dell’utente.
- NavBarPages, mostra le pagine accessibile all’utente in base al ruolo.
- BillModal, visualizza il conto di un determinato tavolo.
- ChangePasswordModal, visualizza una finestra per la modifica della password di un utente.
- CreateUsermodal, visualizza una finestra per la creazione di un nuovo utente.
- TakeOrdersModal, finestra per la gestione degli ordini.
- OccupyTableModal, finestra per la scelta del tavolo in base al numero di clienti.
- Info, una pagina per visualizzare info e credits.
- KitchenPage, una pagina per gestire sia la cucina che il bar, permette la visualizzazione e la gestione degli ordini in base all’utente che ha effettuato l’accesso.
- Login, pagina per effettuare l’accesso.
- TablesPage, pagina per la gestione dei tavoli.

- UsersPage, pagina per la gestione degli utenti.
- WaiterTablesPage, pagina per la gestione dei tavoli da parte del cameriere.
- LiveOrders, template per la gestione in tempo reale degli ordini.
- LiveTables, template per la gestione in tempo reale dei tavoli.
- OrderContent, template per la visualizzazione dei dettagli di un ordine.
- TableContent, template per la visualizzazione dei dettagli di un tavolo.
- UserContent, template per la visualizzazione dei dettagli di un utente.
- MenuItemContent, template per la visualizzazione dei dettagli di un menuItem.

I servizi sono:

- AuthGuardService, usato per garantire o negare l'accesso a certe route agli utenti. Controlla il token ed il ruolo dell'utente.
- AuthService, usato per la gestione del token di autenticazione, come salvataggio, lettura e cancellazione.
- EventsService, usato per ascoltare gli eventi lato server gestiti tramite socket.io.
- LoginService, usato per inviare la richiesta di login al server.
- MenuItemsService, mette a disposizione funzioni per accedere agli endpoint di menu.
- NoAuthGuardService, usato per controllare che l'utente non sia autenticato prima di accedere alla pagina di login.
- OrdersService, mette a disposizione funzioni per accedere agli endpoint di orders.
- RootGuardService, usato per determinare in quale route deve essere rediretto l'utente, ad esempio il cuoco viene rediretto verso la pagina dedicata alla cucina.
- TablesService, mette a disposizione funzioni per accedere agli endpoint di tables.

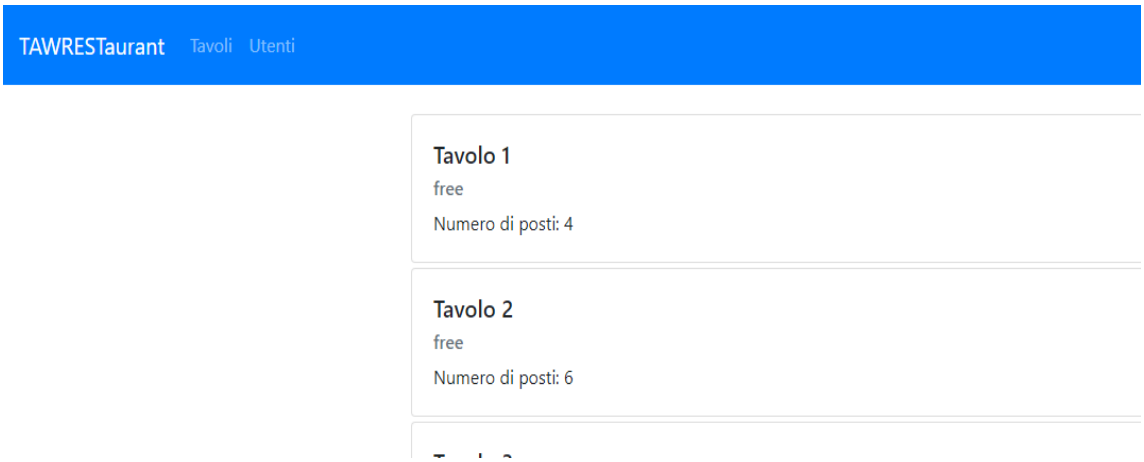
- UsersService, mette a disposizione funzioni per accedere agli endpoint di users.

Le routes sono:

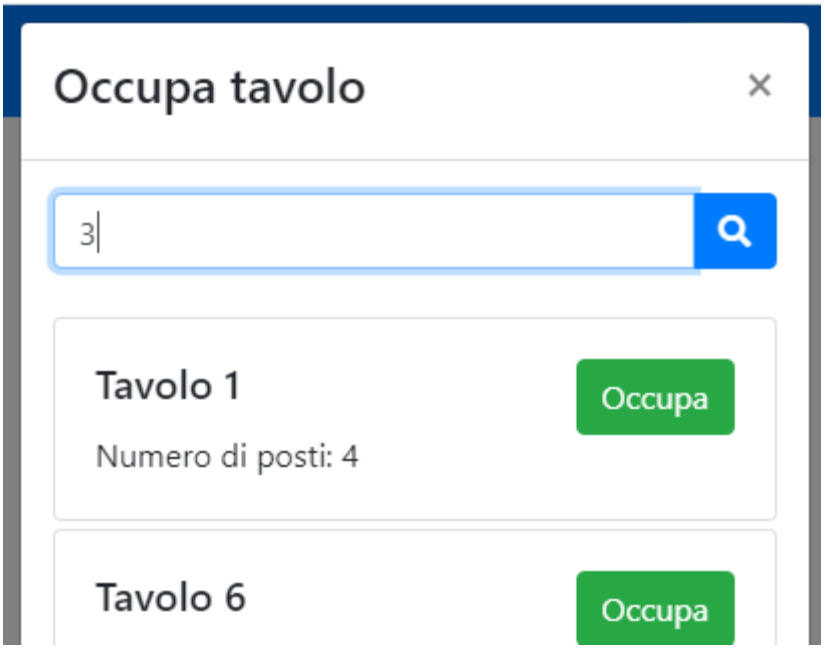
- root, porta utente nella pagina di competenza, login se l'utente non è autenticato.
- login, pagina di login.
- info, pagina con le info e credits del progetto.
- waiter/tables, pagina con l'interfaccia del cameriere per gestire nuovi clienti, tavoli e ordini.
- cook/kitchen, pagina per gestire gli ordini in arrivo.
- barman/bar, pagina per gestire gli ordini in arrivo.
- cashier/tables, pagina per supervisionare lo stato dei tavoli e per generare il conto.
- cashier/users, pagina per la gestione degli utenti.

Esempi di workflow tipico dell'applicazione:

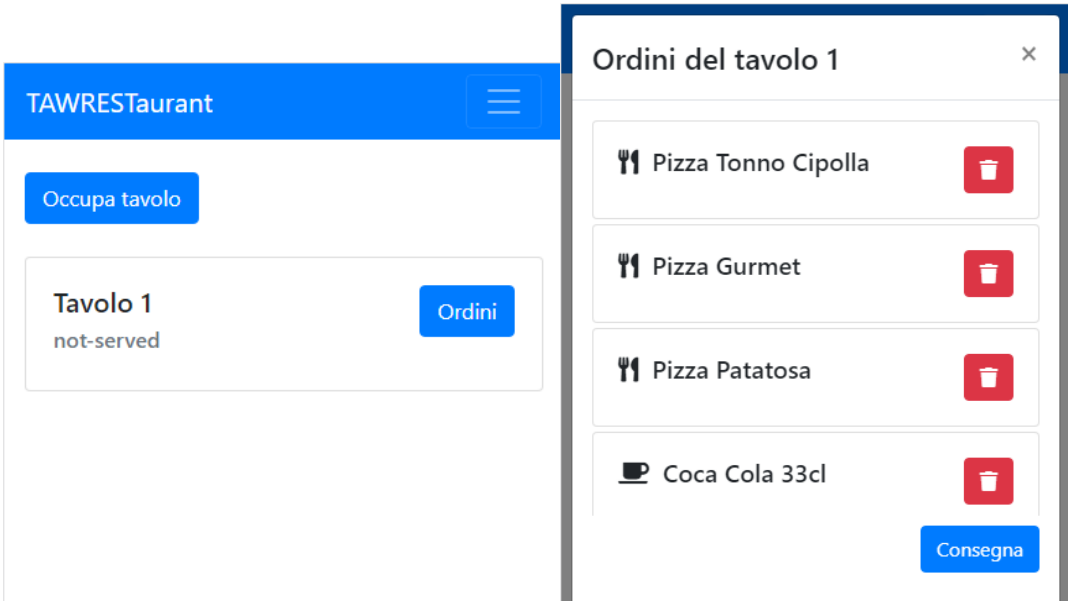
1. Stato iniziale: Nessun tavolo occupato, nessun ordine in preparazione.



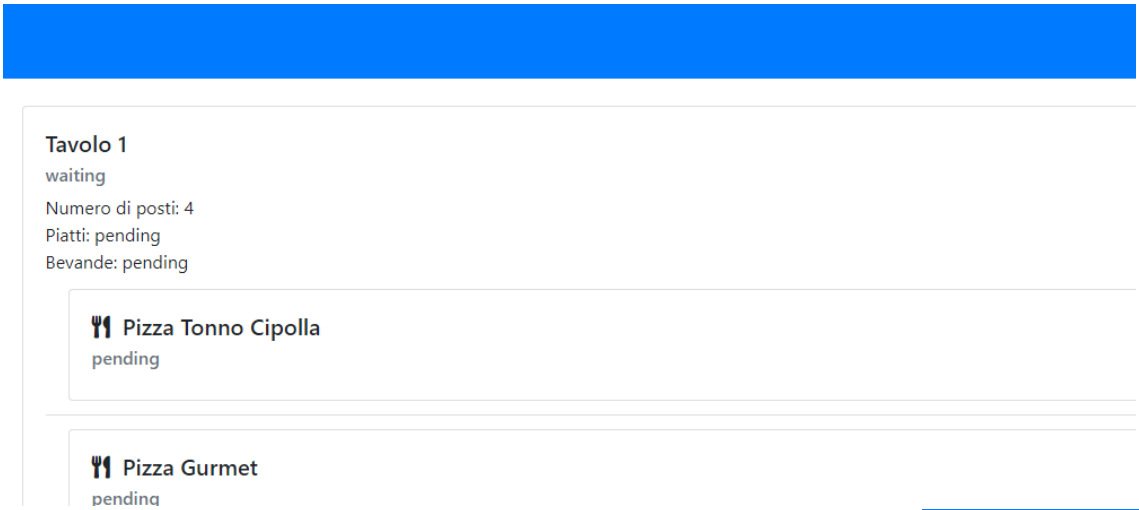
2. 3 clienti entrano nel ristorante e vengono accolti da un cameriere.
3. Con l'app da smartphone il cameriere cerca un tavolo per 3 clienti, lo trova e lo occupa.



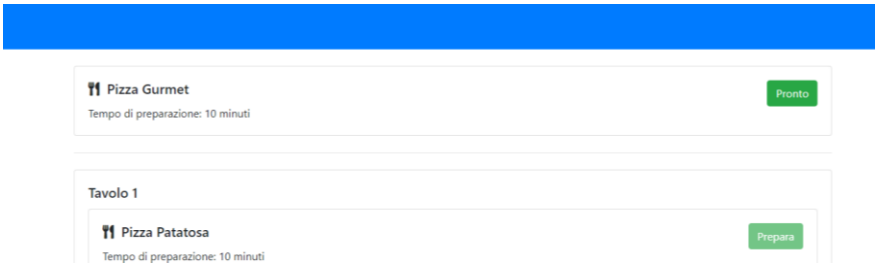
4. Fa sedere i clienti al tavolo.
5. Dopo aver aspettato che i clienti leggano il menu il cameriere torna al tavolo.
6. Il cameriere apre la finestra degli ordini del tavolo da lui servito. Accetta gli ordini di bevande e alimenti assieme e li invia alla cucina e al bar.

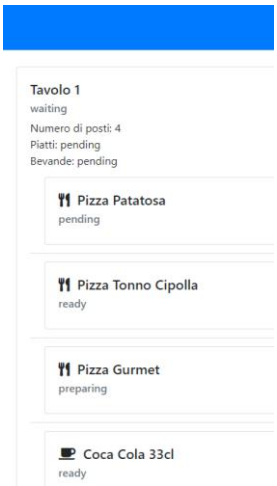


7. Il cassiere ha potuto vedere il cambio di stato del tavolo da libero ad occupato, e dopo che il cameriere ha inviato gli ordini in cucina può vedere lo stato dei singoli ordini.



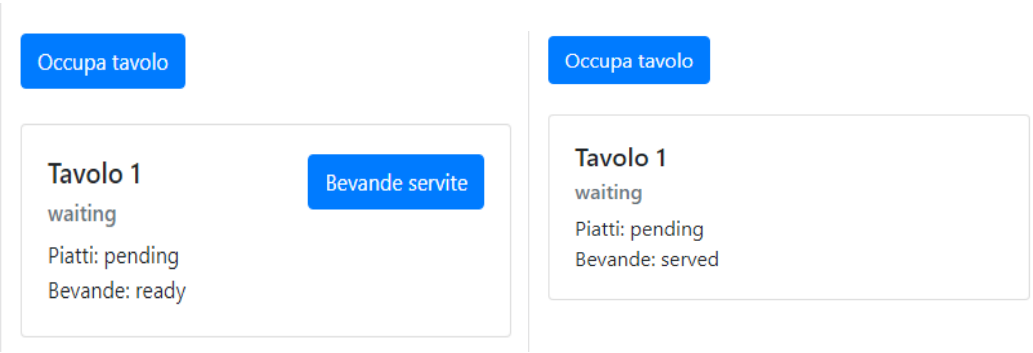
8. Ora il barista e il cuoco iniziano a preparare i rispettivi ordini. Il cassiere può vedere in tempo reale ogni cambiamento.





9. Una volta che il barista ha preso un ordine e lo ha preparato lo notifica. Dopo aver preparato e notificato tutte le bevande il

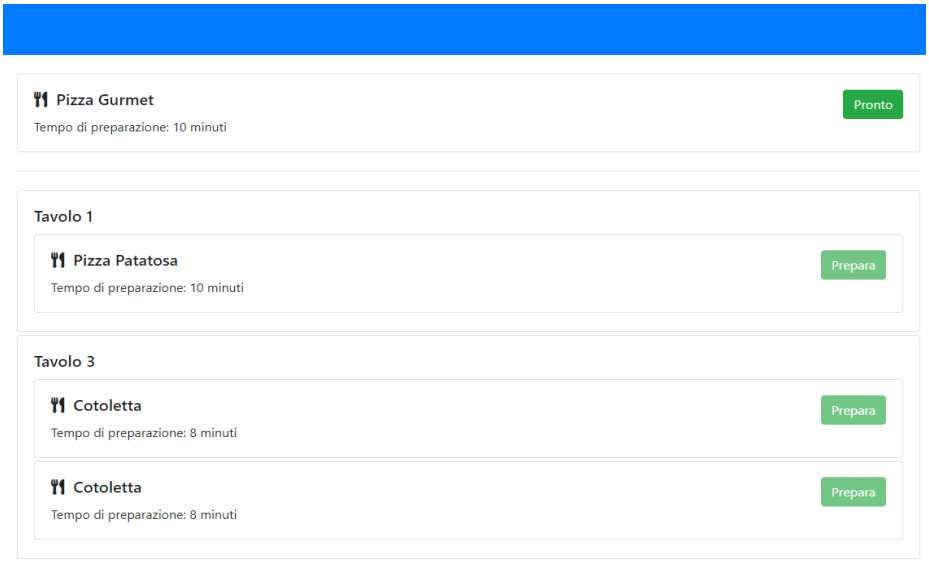
cameriere viene notificato che tutte le bevande sono pronte e le porta al tavolo. Il cameriere segnala che le bevande sono state servite.



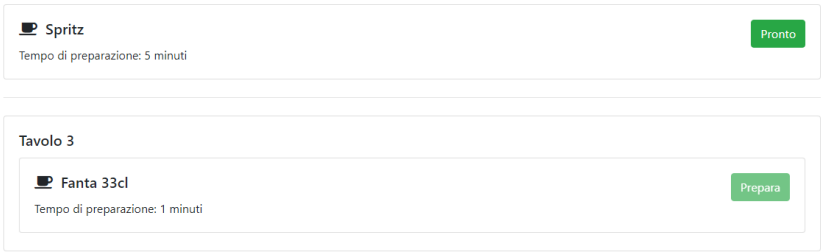
10. Arrivano altri clienti, un secondo cameriere li accoglie e li fa sedere su un tavolo dopo averlo scelto dal proprio smartphone. Inoltra gli ordini e attende.

11. Il cassiere vede che un altro tavolo è stato occupato e che il primo ha le bevande servite, ma gli alimenti sono in preparazione.

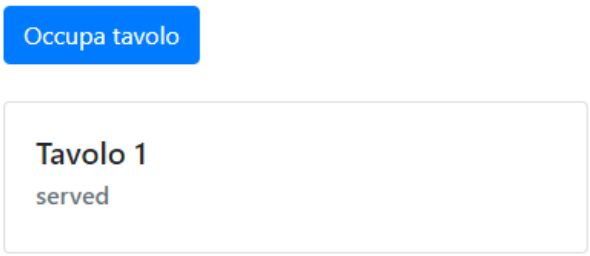
12. In cucina e arrivano le ordinazioni del nuovo tavolo, ma devono prima finire di servire il primo.



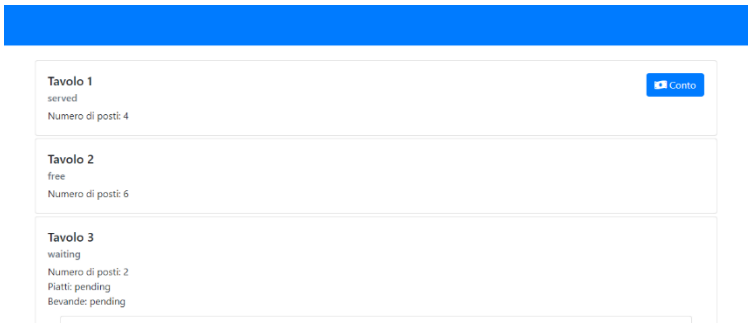
13. Il barista può iniziare a servire il secondo tavolo.

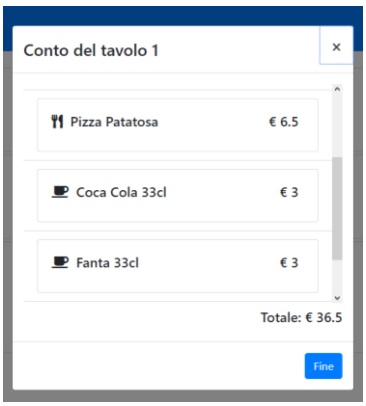


14. I cuochi hanno finito di preparare gli ordini del primo tavolo, il cameriere serve i piatti e il cassiere ha il primo tavolo servito e il secondo in attesa.

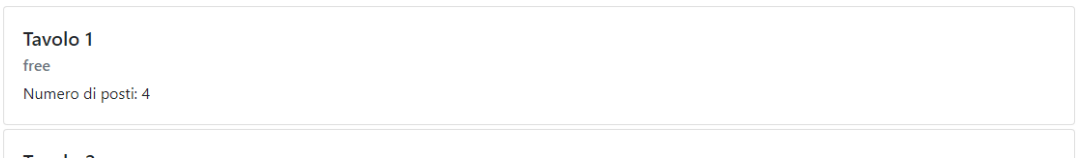


15. I clienti al primo tavolo hanno finito di mangiare e vanno verso il cassiere, il quale calcola il conto e lo porge ai clienti.





16. I clienti pagano e il tavolo viene liberato.



17. A questo punto i cuochi, baristi e camerieri hanno le statistiche aggiornate.