

TECNOLOGIE E APPLICAZIONI WEB

WATTENE

APPLICAZIONE PER LA GESTIONE DI UN PSEUDO-RISTORANTE



INDICE

| | |
|--------------------|---------|
| INTRODUZIONE | pag. 3 |
| GESTIONE DEI RUOLI | pag. 4 |
| FUNZIONALITÀ | pag. 5 |
| BACK-END | pag. 7 |
| ENDPOINTS | pag. 10 |
| AUTENTICAZIONE | pag. 13 |
| ANGULAR | pag. 14 |
| WORKFLOW | pag. 17 |

INTRODUZIONE

L'applicazione Wattene è un'applicazione web che consente la gestione di un ristorante. L'applicazione è costituita da un back-end, scritto in linguaggio Typescript per eseguirlo nell'ambiente Node.js e utilizza MongoDB come DBMS per la persistenza dei dati, e da un front-end, scritto in Angular nella versione 8.2.14 e si utilizzano Apache Cordova, per utilizzare l'applicazione in formato app mobile, ed Electron, per utilizzare l'applicazione in ambiente desktop come applicazione nativa.

Viene utilizzato Socket.io per la comunicazione in tempo reale tra front-end e back-end ed Express.js come middleware per la gestione del routing

GESTIONE DEI RUOLI

L'applicazione distingue gli utenti in base al ruolo di appartenenza per presentare pagine e funzionalità diverse per ognuno di essi.

Gli utenti definiti nell'applicazione sono: cassa, cameriere, cuoco e barista.

Cassa

1. Gestisce l'occupazione dei tavoli;
2. Effettua il pagamento delle ordinazioni dei tavoli;
3. Creazione e gestione dei tavoli, utenti, cibi e bevande;
4. Visualizza le statistiche sullo stato degli ordini;
5. Conoscere il guadagno totale di una giornata.

Cameriere

1. Prende e consegna le ordinazioni dei tavoli a cui è stato assegnato.

Cuoco

1. Prende le ordinazioni in tempo reale dai camerieri sui cibi da preparare;
2. Notifica il cameriere per l'inizio e la fine della preparazione di un ordine.

Barista

1. Prende le ordinazioni in tempo reale dai camerieri sulle bibite da preparare;
2. Notifica il cameriere per l'inizio e la fine della preparazione di un ordine.

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.

2. Gestione dei tavoli e degli ordini per i camerieri

- a. Modifica dello stato di un tavolo da libero a occupato da un certo numero di clienti;
- b. Aggiunta di una o più ordinazioni di piatti o/e bevande per un determinato tavolo. Quando le ordinazioni sono ultimate queste vengono notificate ai cuochi o/e baristi;
- c. 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 per i cuochi

- a. Visualizzazione in tempo reale della coda di preparazione dei cibi per ciascun tavolo con la logica FIFO, dove i cibi da preparare vengono ordinati per tempo di preparazione, dal più lungo al più breve;
- b. Il cuoco notifica il cameriere per l'inizio e la fine della preparazione dei cibi;
- c. Una volta che tutti i cibi per un determinato tavolo sono stati preparati, il cameriere addetto a quel tavolo viene notificato per procedere al servizio.

4. Gestione della coda di preparazione delle bevande per i baristi

-
- a. Visualizzazione in tempo reale della coda di preparazione delle bevande per ciascun tavolo;
 - b. Una volta che tutte le bevande per un determinato tavolo sono stati preparati, il cameriere addetto a quel tavolo viene notificato per procedere al servizio.

5. Gestione cassa per i cassieri

- a. Calcolo del costo 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.).

BACK-END

Il back-end è un server che mette a disposizione delle API in stile RESTful.

Le tecnologie utilizzate sono:

- Node.js: è un framework che consente di scrivere applicazioni in JavaScript lato server;
- Express.js: è un framework per applicazioni web Node.js di routing e middleware;
- Mongoose.js: è una libreria per l'Object data Modeling (ODM) per MongoDB e Node.js. Gestisce le relazioni tra i dati e traduce gli oggetti Javascript e li rappresenta in MongoDB;
- MongoDB: è un database non relazionale dove utilizza i documenti e le collezioni di oggetti;
- Socket.io: è una libreria che permette la comunicazione in tempo reale, bidirezionale e basata su eventi tra il browser ed il server.

MongoDB: struttura documenti e collezioni

Il database contiene 6 collezioni:

1. Users:

- a. `_id`: string; identificativo generato automaticamente da mongoDB;
- b. `username`: string; nome utente univoco;
- c. `firstname`: string; nome utente;
- d. `lastname`: string; cognome utente;
- e. `role`: string; ruolo utente (cassa, cameriere, cuoco, barista);
- f. `hash`: string; password codificata tramite bcryptjs;
- g. `refresh_token`: string; token che verrà utilizzato per sostituire il token scaduto;
- h. `deletion_date`: date; data cancellazione dell'utente

2. Table:

- a. `_id`: string; identificativo generato automaticamente da mongoDB;
- b. `waiter_id`: string; identificativo del cameriere assegnato al tavolo;
- c. `order_id`: string; identificativo dell'ordine del tavolo;
- d. `table_number`: number; numero univoco del tavolo;

-
- e. available: boolean; disponibilità del tavolo;
 - f. total_seats: number; numero dei posti del tavolo;
 - g. clients: number; numero dei clienti seduti sul tavolo;

3. Food:

- a. _id: string; identificativo generato automaticamente da mongoDB;
- b. name: string; nome del cibo;
- c. price: number; prezzo del cibo;
- d. time: number; tempo di preparazione del cibo;
- e. category: string; categoria del cibo (Dessert, Antipasto, Primo piatto, Secondo piatto, Contorno, Dolce);
- f. available: boolean; disponibilità del cibo;

4. Drink:

- a. _id: string; identificativo generato automaticamente da mongoDB;
- b. name: string; nome della bevanda;
- c. price: number; prezzo della bevanda;
- d. time: number; tempo di preparazione della bevanda;
- e. category: string; categoria della bevanda (Alcolico, Analcolico);
- f. available: boolean; disponibilità della bevanda;

5. Order:

- a. _id: string; identificativo generato automaticamente da mongoDB;
- b. waiter_id: string; identificativo del cameriere;
- c. table_id: string; identificativo del tavolo;
- d. paid: boolean; true se l'ordine è stato pagato, false altrimenti
- e. served: boolean; true se l'ordine è stato eseguito, falso altrimenti
- f. amount: number; costo totale dell'ordinazione
- g. clients: number; numero di clienti (serve ai fini statistici)

6. Suborder:

-
- a. id: string; identificativo generato automaticamente da mongoDB;
 - b. order_id: string; identificativo ordine a cui è riferito;
 - c. state_foods: string; stato di preparazione dei cibi (in coda, in preparazione, servito, pagato);
 - d. state_drinks: string; stato di preparazione delle bevande (in coda, in preparazione, servito, pagato);
 - e. ordered_foods?: { food_id, quantity, time, name, price, prepared_by };
 - f. ordered_drinks?: { drink_id, quantity, time, name, price, prepared_by };
 - i. food_id/drink_id: string; identificativo del/la cibo/bevanda;
 - ii. quantity: number; quantità del/la cibo/bevanda;
 - iii. time: number; tempo di preparazione;
 - iv. name: string; nome cibo/bevanda;
 - v. price: number; prezzo;
 - vi. prepared_by: string; identificativo del cuoco/barista che ha preparato il/la cibo/bevanda (serve ai fini statistici);

ENDPOINTS del server a partire dell'endpoint `"/api/v1/"`

| METODO | ENDPOINT | RUOLO | REQUEST | RESPONSE | DESCRIZIONE |
|--------|---------------|---------------------|--|------------------------------------|--|
| GET | | | | JSON {api} | Restituisce l'elenco degli endpoints |
| | USERS | | | | |
| GET | users/ | Cassa | | JSON {user}[] | Restituisce tutti gli utenti |
| GET | users/current | | | JSON {user} | Restituisce l'utente che fa la richiesta |
| GET | users/:id | Cassa | | JSON {user} | Restituisce l'utente tramite il suo id |
| POST | users/ | Cassa | body: username, password, role, lastname, firstname | JSON {user} | Registra un utente |
| POST | users/login | | body: username, password | JSON {user} | Restituisce tutti i dati dell'utente con il refresh_token e senza l'hash |
| PUT | users/:id | Cassa | body: username, password, role, lastname, firstname, refresh_token, deletion_date | JSON {user} | Modifica i dati di un utente (lo username non può essere modificato) |
| DELETE | users/:id | Cassa | | JSON {} | Elimina l'utente dell'utente |
| POST | users/refresh | | body: refresh_token, user_id | JSON { token, refresh_token} | Rinnova il token dell'utente |
| | TABLES | | | | |
| GET | tables/ | | | JSON {table}[] | Ritorna tutti i tavoli |
| GET | tables/:id | | | JSON {table} | Ritorna il tavolo tramite il suo id |
| POST | tables/ | Cassa | body: table_number, total_seats | JSON {table} | Crea un tavolo |
| PUT | tables/:id | Cassa, Cameriere | body: clients, | JSON {table} | Modifica un tavolo in base al suo id |

| | | | | | |
|--------|-------------------|-------|---|-----------------|--|
| | | | waiter_id, available, table_number, order_id, total_seats | | |
| DELETE | tables/:id | Cassa | | JSON {} | Elimina il tavolo in base al suo id |
| | FOODS | | | | |
| GET | foods/ | | | JSON {food}[] | Ritorna tutti i cibi |
| GET | foods/:id | | | JSON {food} | Ritorna il cibo in base al suo id |
| GET | foods/categories | | | JSON {string}[] | Ritorna tutte le categorie dei cibi |
| POST | foods/ | Cassa | body: name, price, time, category, available | JSON {food} | Crea un cibo |
| PUT | foods/:id | Cassa | body: name, price, time, category, available | JSON {food} | Modifica un cibo in base al suo id |
| DELETE | foods/:id | Cassa | | JSON {} | Elimina un cibo in base al suo id |
| | DRINKS | | | | |
| GET | drinks/ | | | JSON {food}[] | Ritorna tutti le bevande |
| GET | drinks/:id | | | JSON {food} | Ritorna la bevanda in base al suo id |
| GET | drinks/categories | | | JSON {string}[] | Ritorna tutte le categorie delle bevande |
| POST | drinks/ | Cassa | body: name, price, time, category, available | JSON {food} | Crea una bevanda |
| PUT | drinks/:id | Cassa | body: name, price, time, category, available | JSON {food} | Modifica una bevanda in base al suo id |

| | | | | | |
|--------|--|---|--|-----------------------|--|
| DELETE | drinks/:id | Cassa | | JSON {} | Elimina una bevanda in base al suo id |
| | ORDERS | | | | |
| GET | orders/ | | | JSON {order}[] | Ritorna tutti gli ordini |
| GET | orders/:id | | | JSON {order} | Ritorna l'ordine in base al suo id |
| GET | orders/:order_id/ suborders | Cassa, Cameriere, Cuoco, Barista | | JSON {suborder}[] | Ritorna tutti i suborder appartenenti ad un ordine |
| GET | orders/suborders | Cassa, Cameriere, Cuoco, Barista | query: type ("food", "drink"), ready | JSON {suborders}[] | Ritorna tutti i suborders che hanno bevande e/o cibi |
| POST | orders/ | Cassa, Cameriere | body: waiter_id, table_id, paid, served, amount, clients | JSON {order} | Crea un nuovo ordine |
| POST | orders/:order_id/ suborders | Cassa, Cameriere | | JSON {suborder} | Crea un suborder di un tavolo (aggiunge cibi o bevande all'ordine) |
| PUT | orders/:order_id/ suborders/ :suborder_id/foods/ :food_id | Cuoco, Cameriere | body: ready | JSON {suborder} | Modifica lo stato di ready di un cibo di un suborder |
| PUT | orders/:order_id/ suborders/ :suborder_id/drinks/ :drink_id | Barista, Cameriere | body: ready | JSON {suborder} | Modifica lo stato di ready di una bevanda di un suborder |
| PUT | orders/:order_id/ suborders/ :suborder_id | Cassa, Cameriere | body: food_served, drink_served | JSON {suborder} | Modifica il campo state_foods/state_drinks |
| PUT | orders/:id | Cassa, Cameriere | body: waiter_id, table_id, paid, served, amount, clients | JSON {order} | Modifica un ordine |
| DELETE | orders/:id | Cassa | | JSON {} | Elimina un ordine |

AUTENTICAZIONE

Per autenticarsi, l'utente dovrà inserire le proprie credenziali nell'apposito form nella pagina di login. L'hash della password viene confrontato con quello presente nel database e viene generato un token (JWT) associato all'utente in caso di successo. L'api ritornerà i dati dell'utente (tranne la sua password) dove verrà popolato il campo "refresh_token" che verrà usato per sostituire il token quando questo sarà scaduto.

L'autenticazione in fase di login non è sicura poiché l'applicativo utilizza il protocollo http e non https. Inoltre se il token venisse rubato, questo potrebbe essere utilizzato per accedere al profilo dell'utente a sua insaputa.

Per l'autenticazione viene utilizzato expressJWT, un middleware che si gestisce la validazione dei JsonWebTokens.

ANGULAR

Angular è un framework open source per lo sviluppo di applicazioni web.

Per lo stile delle pagine abbiamo utilizzato angular-material.

L'applicazione presenta pagine differenti in base al ruolo dell'utente dove:

- il cassiere ha il ruolo di amministratore e gestisce i cibi, le bevande, gli utenti, i tavoli e i pagamenti;
- il cameriere occupa i tavoli, crea le ordinazioni notificando i cuochi e/o i baristi e li consegnano non appena sono pronti;
- il cuoco e il barista prepara le ordinazioni e notificano il cameriere quando questi sono pronti.

I component dell'applicazione sono:

1. **Navbar:** barra di navigazione presente in tutte le pagine;
2. **Login:** pagina di login per effettuare l'accesso;
3. **Dashboard:** pagina che compare dopo aver fatto il login e varia in base al ruolo dell'utente: la cassa e i camerieri hanno la lista dei tavoli, i baristi e i cuochi hanno la lista degli ordini da completare e i camerieri hanno anche la lista delle ordinazioni pronte per essere servite ai tavoli;
 - La dashboard contiene:
 - a. **Serve-queue:** lista delle ordinazioni pronte per essere consegnate dai camerieri;
 - b. **Preparation-queue:** lista delle ordinazioni da preparare per i cuochi e per i baristi;
 - c. **Dialog-payment:** finestra di dialogo dove compare uno scontrino con tutti i cibi e le bevande;
 - d. **Dialog-table:** finestra di dialogo per la creazione e per l'occupazione dei tavoli;
4. **Order:** pagina su cui il cameriere prende le ordinazioni, sono presenti le liste dei cibi e delle bevande;

-
5. **Backoffice:** pagina di gestione degli utenti, dei cibi e delle bevande e controllo delle statistiche per la cassa;
- Il backoffice contiene:
 - a. **Register:** form per la creazione degli utenti;
 - b. **Food:** tabella per la creazione e per la gestione dei cibi;
 - c. **Drink:** tabella per la creazione e per la gestione delle bevande;
 - d. **Users:** tabella per la gestione degli utenti;
 - e. **Stats:**
 - i. **Barman-stats:** statistiche dei baristi;
 - ii. **Cook-stats:** statistiche dei cuochi;
 - iii. **drink-stats:** statistiche delle bevande;
 - iv. **food-stats:** statistiche dei cibi;
 - v. **money-stats:** statistiche sui guadagni;
 - vi. **waiter-stats:** statistiche dei camerieri.

I servizi dell'applicazione sono:

1. **Auth-guard:** servizio che verifica il token sia valido, se il token risulta non valido, lo rimanda alla pagina di login;
2. **No-auth-guard:** servizio che verifica se l'utente si è autenticato, se risulta autenticato, lo reindirizza nella pagina /dashboard dalla pagina di login;
3. **Role-guard:** servizio che verifica se l'utente si è autenticato e se presenta il ruolo corretto, se una delle due condizioni risulta negativa, l'utente viene reindirizzato nella pagina di login;
4. **Req-interceptor:** servizio che intercetta le richieste http, aggiunge il bearer token all'header e in caso di errore 401 prova a rinnovare il token, se ritorna nuovamente un errore 404, l'utente viene reindirizzato alla pagina di login;
5. **SocketIO:** servizio che utilizza i socket per mandare le notifiche ai vari componenti che si mettono in ascolto;
6. **User:** servizio che mette a disposizione le funzioni per accedere agli endpoints /users;
7. **Table:** servizio che mette a disposizione le funzioni per accedere agli endpoints /tables;
8. **Food:** servizio che mette a disposizione le funzioni per accedere agli endpoints /foods;
9. **Drink:** servizio che mette a disposizione le funzioni per accedere agli endpoints /drinks;
10. **Order:** servizio che mette a disposizione le funzioni per accedere agli endpoints /orders.

Le routes sono:

1. **""**: path del root, reindirizza automaticamente al path **"/dashboard"** se l'utente è autenticato, altrimenti va alla pagina di login;
2. **"login"**: path della pagina di login, se l'utente risulta autenticato, esso viene reindirizzato nella dashboard;
3. **"dashboard"**: path della pagina dashboard, se l'utente non è autenticato, esso viene reindirizzato nella pagina di login;
4. **"backoffice"**: path della pagina backoffice, se l'utente non è autenticato o se non ha il ruolo di cassa, esso viene reindirizzato nella pagina di login;
5. **"orders/:id"**: path della pagina dell'ordine, se l'utente non è autenticato o se non ha il ruolo di cameriere, esso viene reindirizzato nella pagina di login.

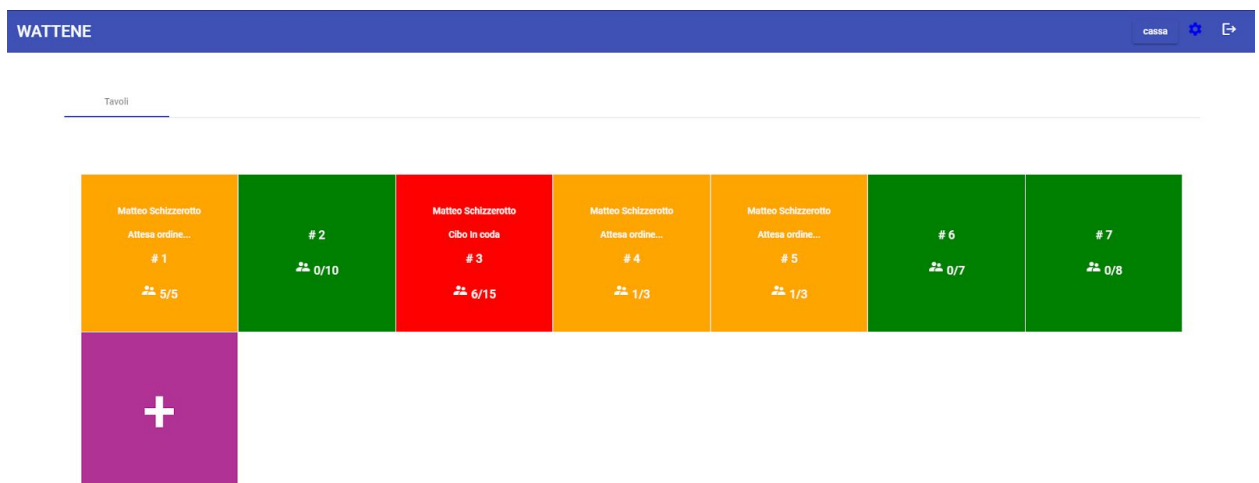
WORKFLOW

1. L'utente accede dalla pagina di login con le proprie credenziali



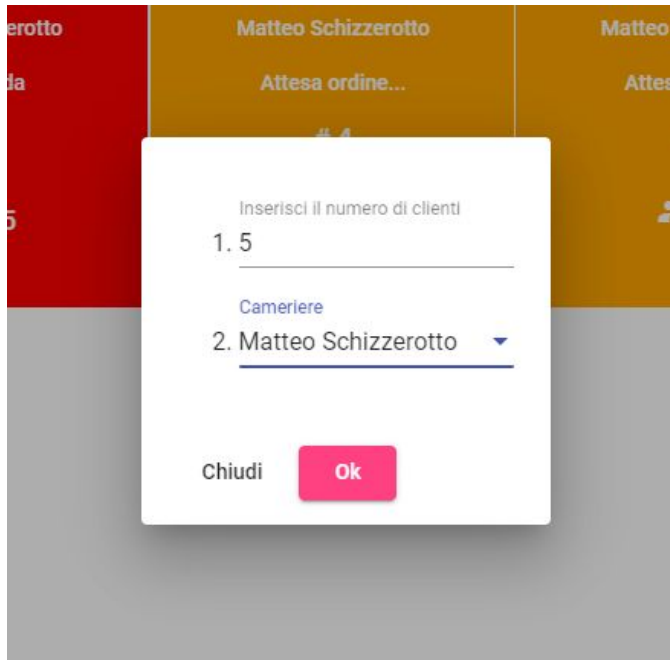
A screenshot of a login form titled "LOGIN". It features two input fields: "Username" with the text "cassa" and "Password" with masked characters "*****". Below the fields is a blue button labeled "Conferma".

2. Il cassiere avrà a disposizione una lista di tutti i tavoli e le informazioni necessarie per la loro gestione nella dashboard.

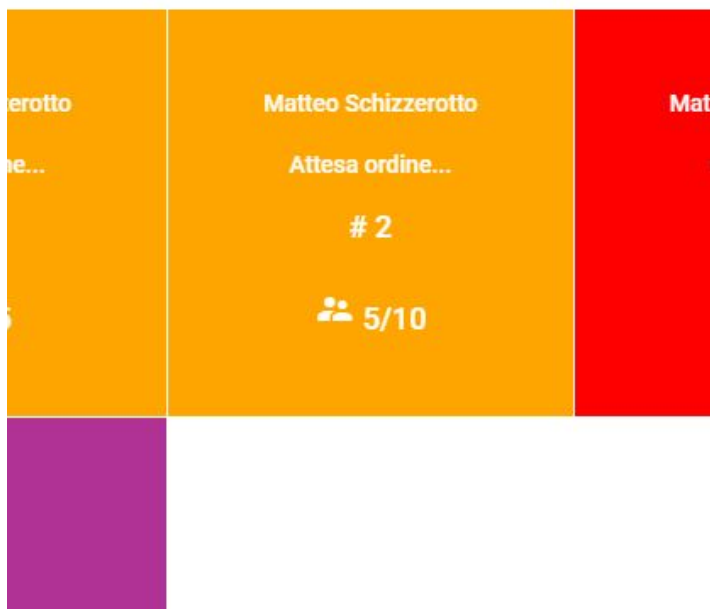


A screenshot of a dashboard titled "WATTENE". The top navigation bar is blue with the name "WATTENE" on the left and a user profile "CASSA" with a settings gear and a home icon on the right. Below the navigation bar, there is a section titled "Tavoli" (Tables). It displays a grid of table status cards. The first row contains seven cards: an orange card for table #1 (Matteo Schizzerotto, Attesa ordine..., 5/5), a green card for table #2 (0/10), a red card for table #3 (Matteo Schizzerotto, Cibo in coda, 6/15), an orange card for table #4 (Matteo Schizzerotto, Attesa ordine..., 1/3), an orange card for table #5 (Matteo Schizzerotto, Attesa ordine..., 1/3), a green card for table #6 (0/7), and a green card for table #7 (0/8). Below the first card is a purple card with a white plus sign, indicating a new table can be added.

3. La cassa clicca su un tavolo e seleziona "Occupa tavolo", successivamente inserisce il numero dei clienti e assegna il cameriere al tavolo.



4. Ora il tavolo risulterà occupato e in attesa di un ordine. Inoltre sarà visibile il nome e il cognome del cameriere assegnato a quel tavolo.



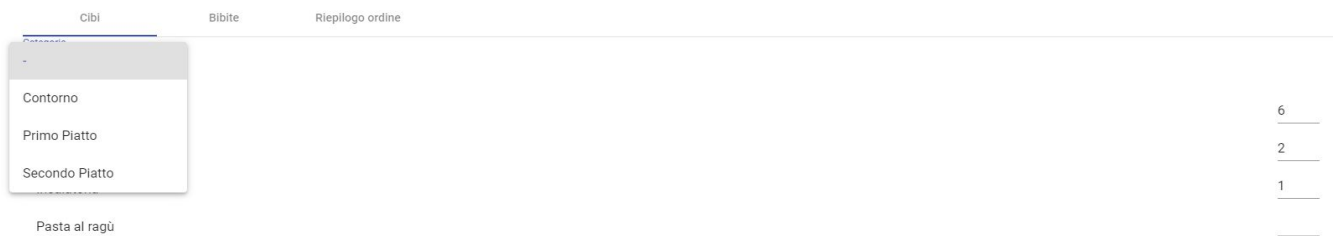
-
5. Il cameriere assegnato al tavolo 2, dalla sua interfaccia dashboard, vedrà ora il tavolo occupato in attesa di un ordine.



-
6. Il cameriere per prendere l'ordinazione del tavolo clicca sopra al tavolo e seleziona "Gestione ordine".



7. Il cameriere ora può selezionare una categoria sui cibi e sulle bevande e la loro quantità nella pagina degli ordini.



8. Successivamente il cameriere può rivedere il riepilogo dell'ordine e confermare l'ordinazione.

| Cibi | Bibite | Riepilogo ordine |
|------------------------------|--------|------------------|
| x6 Cotoletta e patate fritte | | |
| x2 Fagioli in salsa | | |
| x1 Insalatona | | |
| x3 Acqua frizzante | | |
| x2 Sprite | | |
| x1 Tè alla pesca | | |

9. Il barista, dalla sua dashboard, vedrà la lista delle bevande da preparare. Una volta preparate cliccherà sopra alla bevanda per segnare che è stato preparato. Quando avrà preparato tutte le bevande dell'ordinazione, potrà cliccare il tasto pronto per notificare il cameriere.

WATTENEbarista

Ordini

Pronto

Acqua frizzante
x2 Tempo totale: 3 min

Sprite
x2 Tempo totale: 2 min

Tè alla pesca
x1 Tempo totale: 1 min

10. Analogamente per il cuoco.

WATTENEcuoco

Ordini

Pronto

Cotoletta e patate fritte
x6 Tempo totale: 90 min

Fagioli in salsa
x2 Tempo totale: 8 min

Insalatona
x1 Tempo totale: 1 min

11. Ora il cameriere vedrà nella scheda ordini della sua dashboard i cibi e le bevande da servire al tavolo 2. Se ci clicca sopra, il cameriere lo segnerà come consegnato.

Ordini

Cibo del tavolo 2

Drink del tavolo 2

12. La cassa vedrà nella sua dashboard che il cibo e le bevande sono servite al tavolo 2.



13. La cassa potrà procedere al pagamento (che includerà la liberazione del tavolo) cliccando sul tavolo e selezionando paga e libera tavolo.



14. Comparirà un riepilogo dell'ordine del tavolo con la lista dei cibi e delle bevande ordinate con il conto totale da pagare.



15. Una volta pagato, il tavolo 2 ritornerà ad essere libero.



16. Le statistiche vengono ora aggiornate nel backoffice.

