



Università Ca' Foscari Venezia

Dipartimento di Scienze Ambientali, Informatica e Statistica

Corso di Tecnologie e Applicazioni Web - A.A. 2018-2019

Docente: Prof. Filippo Bergamaschi

Relazione riassuntiva

Matteo Schizzerotto, 867704

Data: 17/02/2020

Versione 1.1

Gruppo: WATTENE

867704 Matteo Schizzerotto
870945 Paolo Porcellato
870592 Goran Gajic

Indice

1. Introduzione	3
1.1 Realizzazione	3
1.2 Tecnologia utilizzate	3
2. Modello dei dati	4
2.1 drinks	4
2.2 foods	4
2.3 orders	6
2.4 tables	6
2.5 suborders	7
2.5.1 orderedFoodSchema	7
2.5.2 orderedDrinkSchema	7
2.6 users	8
3. Ruoli	8
3. API	9
3.1 Drink	9
3.2 Cibi	12
3.3 Tavoli	15
3.4 Ordini	18
3.5 Utenti	29
4. Autenticazione	33
5. Frontend	34
5.1 Components	34
5.2 Services	34
5.3 Routes	35
6. Utilizzo	36
6.1 Autenticazione	36
6.2 Gestione utenti	37
6.3 Gestione ordini	38
6.4 Gestione tavoli	41
6.5 Gestione cibi e bevande	41
6.6 Gestione statistiche	42
7. Problemi da tenere in considerazione	43

1. Introduzione

Il progetto di Tecnologie e Applicazioni Web consiste nella realizzazione di un sistema che simula uno pseudo-ristorante e la sua gestione di ordini e tavoli.

1.1 Realizzazione

La realizzazione consiste nella creazione di:

- backend basato su
 - Node.js
 - Express
 - MongoDB
 - API di tipo REST
- tre applicazioni frontend basate rispettivamente su:
 - Angular
 - Apache Cordova
 - Electron

Oltre alla comunicazione tra backend e frontend tramite le API il server offre la comunicazione di messaggi in tempo reale grazie a socket.io, questo permette un aggiornamento dei dati visualizzati per tutte le piattaforme, senza l'utilizzo continuo di request GET.

1.2 Tecnologia utilizzate

- MongoDB: un DBMS non relazionale orientato ai documenti (per la gestione della persistenza dei dati);
- Node.js: un interprete multiplatforma per l'esecuzione di codice Javascript lato server (per lo sviluppo del servizio web);
- Express: un framework per la realizzazione di applicazioni web e API per Node.js (per la gestione del routing del servizio web);
- Socket.io: una libreria per la gestione di eventi in tempo reale tramite la tecnologia WebSocket (per la visualizzazione in tempo reale dello stato dei tavoli e degli ordini);
- JSON Web Token (RFC 7519): uno standard per l'autenticazione stateless tra due attori (per l'autenticazione degli utenti nel sistema);
- Angular: un framework per lo sviluppo di Single-Page Application (per lo sviluppo dell'applicazione web);
- Apache Cordova: un framework per lo sviluppo di applicazioni mobile ibride utilizzando tecnologie web (per lo sviluppo dell'applicazione mobile);
- Electron: un framework per lo sviluppo di applicazioni desktop utilizzando tecnologie web (per lo sviluppo dell'applicazione desktop).

2. Modello dei dati

Il database è composto da sei collezioni: drinks, foods, orders, suborders, tables, users. Nel nostro progetto il database utilizza MongoDB e risiede in un server MongoDB Atlas. Questo ci permette di eseguire il backend in qualsiasi posizione e riuscire comunque ad accedere ad un unico database.

2.1 drinks

Chiave	Tipo	Descrizione
name	string	Il nome univoco del drink. È un valore unico tra tutti i documenti della collection
price	number	Il prezzo del drink
time	number	Tempo richiesto per la preparazione
category	string: enum "Alcolico" "Analcolico"	Tipologia di drink
available	boolean	Disponibilità del drink

2.2 foods

Chiave	Tipo	Descrizione
name	string	Il nome univoco del cibo È un valore unico tra tutti i documenti della collection
price	number	Il prezzo del cibo
time	number	Tempo richiesto per la preparazione
category	string: enum "Dessert" "Antipasto" "Primo Piatto" "Secondo Piatto" "Contorno" "Dolce"	Tipologia di cibo
available	boolean	Disponibilità del cibo

2.3 orders

Chiave	Tipo	Descrizione
waiter_id	ObjectId	Id del cameriere che sta gestendo l'ordine
table_id	ObjectId	Id del tavolo legato all'ordine
paid	boolean	Rappresenta se l'ordine è stato pagato o meno
served	boolean	Rappresenta se l'ordine è stato servito o meno
amount	number	Valore totale dell'ordine
clients	number	Numero di clienti seduti al tavolo dell'ordine

2.4 tables

Chiave	Tipo	Descrizione
waiter_id	ObjectId	Id del cameriere che sta gestendo il tavolo
order_id	ObjectId	Indica l'ordine associato al tavolo
table_number	number	Numero del tavolo
available	boolean	Indica la disponibilità del tavolo
total_seats	number	Numero di posti del tavolo
clients	number	Numero di clienti seduti al tavolo

2.5 suborders

Chiave	Tipo	Descrizione
order_id	ObjectId	Indica l'ordine principale associato
order_foods	orderedFoodSchema[]	Array di cibi ordinati
order_drinks	orderedDrinkSchema[]	Array di drink ordinati
state_foods	string: enum "In coda" "In preparazione" "Servito" "Pagato"	Rappresenta lo stato del sottordine di cibo
state_drinks	string: enum "In coda" "In preparazione" "Servito" "Pagato"	Rappresenta lo stato del sottordine di drink

2.5.1 orderedFoodSchema

Chiave	Tipo	Descrizione
food_id	ObjectId	Id univoco del cibo ordinato
quantity	number	Numero di unità ordinate
time	number	Tempo richiesto per la preparazione
name	string	Nome del cibo ordinato
price	number	Prezzo del cibo ordinato
prepared_by	ObjectId	Id univoco del cuoco che ha preparato il cibo

2.5.2 orderedDrinkSchema

Chiave	Tipo	Descrizione
drink_id	ObjectId	Id univoco del drink ordinato
quantity	number	Numero di unità ordinate
time	number	Tempo richiesto per la preparazione
name	string	Nome del drink ordinato
price	number	Prezzo del drink ordinato
prepared_by	ObjectId	Id univoco del barman che ha preparato il drink

2.6 users

Chiave	Tipo	Descrizione
username	string	Username univoco dell'utente
hash	string	Hash della password dell'utente
firstname	string	Nome dell'utente
lastname	string	Cognome dell'utente
role	string: enum "cameriere" "cuoco" "barista" "cassa"	Ruolo dell'utente
delation_date	Date	Data di cancellazione dell'account
refresh_token	string	Informazione necessaria per refreshare il token

3. Ruoli

All'interno del sistema sono presenti 4 tipi di utente:

- cassa
- cameriere
- barista
- cuoco

Ogni utente ha accesso unicamente alla risorse ad esso necessarie per lo svolgimento della sua mansione.

Vi sono controlli sia tramite API, che non restituiscono dati qualora l'utente che cerca di fare le operazioni non abbia i permessi necessari, sia tramite il frontend con un sistema di guard per le varie sezioni dell'interfaccia.

4. API

In questa sezione sono descritte le API (v1.1) del servizio backend.

Endpoint	/
Metodo	GET
Descrizione	Restituisce la lista di tutti gli endpoint

4.1 Drink

Endpoint	/drinks
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "name": string, "price": number, "time": number, "category": string, "available": boolean }</pre>
Descrizione	Restituisce la lista di tutti i drink

Endpoint	/drinks/categories
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "Categoria 1", "Categoria 2", "..." }</pre>
Descrizione	Restituisce le categorie di drink

Endpoint	/drinks/:id
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "name": string, "price": number, "time": number, "category": string, "available": boolean }</pre>
Descrizione	Restituisce il drink che corrisponde all'id :id

Endpoint	/drinks
Metodo	POST
Parametri	
Formato dei dati della richiesta	<pre>{ "name": string, "price": number, "time": number, "category": string, "available": boolean }</pre>
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "name": string, "price": number, "time": number, "category": string, "available": boolean }</pre>
Descrizione	Crea un nuovo drink

Endpoint	/drinks/:id
Metodo	PUT
Parametri	
Formato dei dati della richiesta	{ "name": string, "price": number, "time": number, "category": string, "available": boolean }
Formato dei dati della risposta	{ "_id": ObjectId, "name": string, "price": number, "time": number, "category": string, "available": boolean }
Descrizione	Aggiorna i dati del drink corrisponde all'id :id con i dati contenuti nel body

Endpoint	/drinks/:id
Metodo	DELETE
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ }
Descrizione	Elimina il drink corrispondente all'id :id

4.2 Cibi

Endpoint	/foods
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ "_id": ObjectId, "name": string, "price": number, "time": number, "category": string, "available": boolean }]
Descrizione	Restituisce la lista di tutti i cibi

Endpoint	/foods/categories
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ "Categoria 1", "Categoria 2", "..." }
Descrizione	Restituisce le categorie di cibi

Endpoint	/foods/:id
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "name": string, "price": number, "time": number, "category": string, "available": boolean }</pre>
Descrizione	Restituisce il cibo che corrisponde all'id :id

Endpoint	/foods
Metodo	POST
Parametri	
Formato dei dati della richiesta	<pre>{ "name": string, "price": number, "time": number, "category": string, "available": boolean }</pre>
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "name": string, "price": number, "time": number, "category": string, "available": boolean }</pre>
Descrizione	Crea un nuovo cibo

Endpoint	/foods/:id
Metodo	PUT
Parametri	
Formato dei dati della richiesta	{ "name": string, "price": number, "time": number, "category": string, "available": boolean }
Formato dei dati della risposta	{ "_id": ObjectId, "name": string, "price": number, "time": number, "category": string, "available": boolean }
Descrizione	Aggiorna i dati del cibo corrisponde all'id :id con i dati contenuti nel body

Endpoint	/foods/:id
Metodo	DELETE
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ }
Descrizione	Elimina il cibo corrispondente all'id :id

4.3 Tavoli

Endpoint	/tables
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "table_number": number, "waiter_id": ObjectId, "order_id": ObjectId, "total_seats": number "clients": number, "available": boolean, }</pre>
Descrizione	Restituisce la lista di tutti i tavoli

Endpoint	/tables/:id
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "table_number": number, "waiter_id": ObjectId, "order_id": ObjectId, "total_seats": number "clients": number, "available": boolean, }</pre>
Descrizione	Restituisce il tavolo che corrisponde all'id :id

Endpoint	/tables
Metodo	POST
Parametri	
Formato dei dati della richiesta	{ "table_number": number, "total_seats": number "clients": number, "available": boolean, }
Formato dei dati della risposta	{ "_id": ObjectId, "table_number": number, "waiter_id": ObjectId, "order_id": ObjectId, "total_seats": number "clients": number, "available": boolean, }
Descrizione	Crea un nuovo tavolo

Endpoint	/tables/:id
Metodo	PUT
Parametri	
Formato dei dati della richiesta	{ "table_number": number, "waiter_id": ObjectId, "order_id": ObjectId, "total_seats": number "clients": number, "available": boolean, }
Formato dei dati della risposta	{ ":id": ObjectId, "table_number": number, "waiter_id": ObjectId, "order_id": ObjectId, "total_seats": number "clients": number, "available": boolean, }
Descrizione	Aggiorna i dati del tavolo corrisponde all'id :id con i dati contenuti nel body

Endpoint	/tables/:id
Metodo	DELETE
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ }
Descrizione	Elimina il tavolo corrispondente all'id :id

4.4 Ordini

Endpoint	/orders
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "table_id": ObjectId, "waiter_id": ObjectId, "clients": number, "paid": boolean, "served": boolean, "amount": number, "createdAt": Date, "updatedAt": Date } []</pre>
Descrizione	Restituisce la lista di tutti gli ordini

Endpoint	/orders/:id
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "table_id": ObjectId, "waiter_id": ObjectId, "clients": number, "paid": boolean, "served": boolean, "amount": number, "createdAt": Date, "updatedAt": Date }</pre>
Descrizione	Restituisce l'ordine che corrisponde all'id :id

Endpoint	/orders/suborders
Metodo	GET
Parametri	type = food
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre> { "_id": ObjectId, "order_id": ObjectId, "state_foods": string, "state_drinks": string, "ordered_foods": { "_id": ObjectId, "name": string, "food_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId, } [], "ordered_drinks": { "_id": ObjectId, "name": string, "drink_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "createdAt": Date, "updatedAt": Date } [] </pre>
Descrizione	Restituisce tutti i suborder che hanno del cibo al loro interno

Endpoint	/orders/suborders
Metodo	GET
Parametri	type = drink
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "order_id": ObjectId, "state_foods": string, "state_drinks": string, "ordered_foods": { "_id": ObjectId, "name": string, "food_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId, } [], "ordered_drinks": { "_id": ObjectId, "name": string, "drink_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "createdAt": Date, "updatedAt": Date } []</pre>
Descrizione	Restituisce tutti i suborder che hanno dei drink al loro interno

Endpoint	/orders/ order_id /suborders
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "order_id": ObjectId, "state_foods": string, "state_drinks": string, "ordered_foods": { "_id": ObjectId, "name": string, "food_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId, } [], "ordered_drinks": { "_id": ObjectId, "name": string, "drink_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "createdAt": Date, "updatedAt": Date } []</pre>
Descrizione	Restituisce tutti i suborder relativi all'ordine con id :id

Endpoint	/orders
Metodo	POST
Parametri	
Formato dei dati della richiesta	{ "table_id": ObjectId, "waiter_id": ObjectId, "clients": number, "paid": boolean, "served": boolean, "amount": number }
Formato dei dati della risposta	{ "_id": ObjectId, "table_id": ObjectId, "waiter_id": ObjectId, "clients": number, "paid": boolean, "served": boolean, "amount": number, "createdAt": Date, "updatedAt": Date, }
Descrizione	Crea un nuovo ordine

Endpoint	/orders/:order_id/suborders
Metodo	POST
Parametri	
Formato dei dati della richiesta	<pre> { "order_id": ObjectId, "ordered_foods": { "_id": ObjectId, "name": string, "food_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "ordered_drinks": { "_id": ObjectId, "name": string, "drink_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [] }</pre>
Formato dei dati della risposta	<pre> { "_id": ObjectId, "order_id": ObjectId, "state_foods": string, "state_drinks": string, "ordered_foods": { "_id": ObjectId, "name": string, "food_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "ordered_drinks": { "_id": ObjectId, "name": string, "drink_id": ObjectId,</pre>

	<pre> "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "createdAt": Date, "updatedAt": Date } </pre>
Descrizione	Crea un nuovo suborder

Endpoint	/orders/:id
Metodo	PUT
Parametri	
Formato dei dati della richiesta	<pre> { "table_id": ObjectId, "waiter_id": ObjectId, "clients": number, "paid": boolean, "served": boolean, "amount": number, "createdAt": Date, "updatedAt": Date } </pre>
Formato dei dati della risposta	<pre> { "_id": ObjectId, "table_id": ObjectId, "waiter_id": ObjectId, "clients": number, "paid": boolean, "served": boolean, "amount": number, "createdAt": Date, "updatedAt": Date, } </pre>
Descrizione	Aggiorna i dati dell'ordine corrisponde all'id :id con i dati contenuti nel body

Endpoint	/orders/:order_id/suborders/:suborder_id/drinks/:drink_id
Metodo	PUT
Parametri	
Formato dei dati della richiesta	<pre>{ "ready": boolean }</pre>
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "order_id": ObjectId, "state_foods": string, "state_drinks": string, "ordered_foods": { "_id": ObjectId, "name": string, "food_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "ordered_drinks": { "_id": ObjectId, "name": string, "drink_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "createdAt": Date, "updatedAt": Date }</pre>
Descrizione	Indica una bibita corrispondente a drink_id, presente all'interno del suborder corrispondente a suborder_id come servita

Endpoint	/orders/:order_id/suborders/:suborder_id/foods/:food_id
Metodo	PUT
Parametri	
Formato dei dati della richiesta	<pre>{ "ready": boolean, }</pre>
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "order_id": ObjectId, "state_foods": string, "state_drinks": string, "ordered_foods": { "_id": ObjectId, "name": string, "food_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "ordered_drinks": { "_id": ObjectId, "name": string, "drink_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "createdAt": Date, "updatedAt": Date }</pre>
Descrizione	Indica una cibo corrispondente a food_id, presente all'interno del suborder corrispondente a suborder_id come servito

Endpoint	/orders/ order_id /suborders/ suborder_id
Metodo	PUT
Parametri	
Formato dei dati della richiesta	<pre>{ "foods_served": boolean, "drinks_served": boolean }</pre>
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "order_id": ObjectId, "state_foods": string, "state_drinks": string, "ordered_foods": { "_id": ObjectId, "name": string, "food_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "ordered_drinks": { "_id": ObjectId, "name": string, "drink_id": ObjectId, "quantity": number, "time": number, "price": number, "prepared_by": ObjectId } [], "createdAt": Date, "updatedAt": Date }</pre>
Descrizione	Aggiorna lo stato di preparazione di cibi e/o bevande di un suborder corrispondente a suborder_id ed appartenente all'ordine order_id

Endpoint	/orders/:id
Metodo	DELETE
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ }
Descrizione	Elimina l'ordine corrispondente all'id :id

4.5 Utenti

Endpoint	/users
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "username": string, "firstname": string, "lastname": string, "role": string, "refresh_token": string, "delation_date": Data "createdAt": Data, "updatedAt": Data } []</pre>
Descrizione	Restituisce la lista di tutti gli utenti

Endpoint	/users/current
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "username": string, "firstname": string, "lastname": string, "role": string, "refresh_token": string, "delation_date": Data "createdAt": Data, "updatedAt": Data }</pre>
Descrizione	Restituisce l'utente attuale

Endpoint	/users/:id
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "username": string, "firstname": string, "lastname": string, "role": string, "refresh_token": string, "delation_date": Data "createdAt": Data, "updatedAt": Data }</pre>
Descrizione	Restituisce l'utente che corrisponde all'id :id

Endpoint	/users
Metodo	POST
Parametri	
Formato dei dati della richiesta	<pre>{ "username": string, "password": string, "firstname": string, "lastname": string, "role": string, "refresh_token": string, }</pre>
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "username": string, "firstname": string, "lastname": string, "role": string, "refresh_token": string, "delation_date": Data "createdAt": Data, "updatedAt": Data }</pre>
Descrizione	Crea un nuovo utente

Endpoint	/users/login
Metodo	POST
Parametri	
Formato dei dati della richiesta	{ "username": string, "password": string }
Formato dei dati della risposta	{ "_id": ObjectId, "username": string, "firstname": string, "lastname": string, "role": string, "token": string, "refresh_token": string, "delation_date": Data "createdAt": Data, "updatedAt": Data }
Descrizione	Effettua il login

Endpoint	/users/refresh
Metodo	POST
Parametri	
Formato dei dati della richiesta	{ "_id": ObjectId, "refresh_token": string }
Formato dei dati della risposta	{ "token": string, "refresh_token": string }
Descrizione	Refresh del token

Endpoint	/users/:id
Metodo	PUT
Parametri	
Formato dei dati della richiesta	<pre>{ "username": string, "firstname": string, "lastname": string, "password": string, "role": string, "refresh_token": string, "delation_date": Data }</pre>
Formato dei dati della risposta	<pre>{ "_id": ObjectId, "username": string, "firstname": string, "lastname": string, "role": string, "refresh_token": string, "delation_date": Data, "createdAt": Data, "updatedAt": Data }</pre>
Descrizione	Aggiorna i dati dell'utente corrisponde all'id :id con i dati contenuti nel body

Endpoint	/users/:id
Metodo	DELETE
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ }
Descrizione	Elimina l'utente corrispondente all'id :id

5. Autenticazione

L'autenticazione è realizzata grazie a due diversi metodi previsti dal protocollo HTTP, ossia Basic Authentication e Bearer Authentication.

Il primo consiste nel semplice invio della coppia (username, password). Questo metodo viene impiegato in questo progetto solo al momento dell'accesso al sistema.

L'utilizzo di una connessione sicura è necessario per mantenere la confidenzialità dei dati trasmessi tramite Basic Authentication. Il progetto al momento supporta HTTPS solo con le app Electron e Cordova, che si interfacciano ad un backend che utilizza Heroku come host.

Il secondo consiste nell'invio di una stringa, detta token, che deve essere riconosciuta dal server nelle richieste successive alla sua generazione.

Nel progetto viene utilizzato un tipo specifico di token, chiamato JSON Web Token (abbr. JWT), definito dallo standard RFC 7519.

Il JWT è autenticato dal server grazie ad una stringa tenuta segreta, il che permette sia al server che al client di condividere le informazioni relative all'utente attraverso il token in sicurezza, poiché una qualsiasi modifica del token da parte del client lo invaliderebbe. Questa caratteristica di JWT permette di rendere l'autenticazione priva di stato (stateless): il server esegue ogni richiesta in modo indipendente dalle altre.

Uno dei problemi nell'utilizzo di un JWT per l'autenticazione consiste nel fatto che il token è simile ad una password, dunque se venisse rubato sarebbe necessario revocarlo, altrimenti potrebbe compromettere la sicurezza dell'utente a cui è associato. Per questo motivo, un JWT dovrebbe essere sempre trasmesso attraverso una connessione sicura.

In aggiunta al token JWT è presente un sistema per rinnovare il token alla sua scadenza. La durata standard del token è 1 giorno ma può essere rinnovato fino ad un massimo di 5 giorni.

6. Frontend

Il front-end del sistema è stato realizzato grazie al framework Angular.

In questa sezione sono descritti i Components, i Services e le Routes sviluppati per la sua realizzazione.

6.1 Components

I componenti dell'applicazione sono:

- **App**: la radice dell'applicazione, che presenta la barra di navigazione e il componente per la visualizzazione delle pagine in modo dipendente dallo stato corrente del router;
- **Navbar**: barra di navigazione;
- **Login**: pagina di login per effettuare l'accesso;
- **Dashboard**: pagina che compare dopo aver fatto il login e varia in base all'utente. La cassa e i camerieri hanno la lista dei tavoli, i baristi e i cuochi hanno la lista degli ordini da preparare e i camerieri hanno anche la lista delle ordinazioni pronte per essere servite ai tavoli;
 - **Serve-queue**: lista delle ordinazioni pronte per essere consegnate al tavolo da parte dei camerieri;
 - **Preparation-queue**: lista delle ordinazioni da preparare cuochi e baristi;
 - **Dialog-payment**: finestra di dialogo dove compare uno scontrino con tutti i cibi e le bevande acquistati dal tavolo che sta pagando;
 - **Dialog-table**: finestra di dialogo per la creazione e per l'occupazione dei tavoli;
- **Order**: pagina dove cameriere prende le ordinazioni
- **Backoffice**: pagina per la gestione degli utenti, dei cibi e delle bevande e controllo delle statistiche per la cassa;
 - **Register**: form per la creazione del profilo utente;
 - **Food**: viene gestita la creazione e la gestione dei cibi;
 - **Drink**: tabella per la creazione e per la gestione delle bevande;
 - **Users**: tabella per la gestione degli utenti;
 - **Stats**: Lista di tabelle per le statistiche (clienti, bevande e cibi più venduti, camerieri, cuochi e baristi);
 - **Barman-stats**: statistiche dei baristi;
 - **Cook-stats**: statistiche dei cuochi;
 - **Drink-stats**: statistiche sulle vendite di bevande;
 - **Food-stats**: statistiche sulle vendite dei cibi;
 - **Money-stats**: statistiche relative ai guadagni e agli utenti serviti;
 - **Waiter-stats**: statistiche dei camerieri;

6.2 Services

- **auth-guard**: guardia del router che attiva la route radice "/" se l'utente non è autenticato, provocando così il reindirizzamento alla pagina corretta;
- **role-guard**: guardia che verifica se l'utente ha il ruolo corretto per la route a cui vuole accedere, in caso negativo, l'utente viene reindirizzato nella pagina di login;
- **no-auth-guard**: servizio che verifica se l'utente si è autenticato. Se risulta autenticato, lo reindirizza alla pagina /dashboard dalla pagina di login;
- **req-interceptor**: servizio che intercetta le richieste HTTP, aggiunge il Bearer Token all'header e in caso di errore prova a rinnovare il token
- **SocketIO**: servizio che utilizza i socket per mandare le notifiche ai vari componenti che si mettono in ascolto;
- **users**: servizio che presenta i metodi per la gestione degli utenti;
- **tables**: servizio che presenta i metodi per la gestione dei tavoli;
- **orders**: servizio che presenta i metodi per la gestione delle ordinazioni;
- **food**: servizio che presenta i metodi per la gestione dei cibi;
- **drink**: servizio che presenta i metodi per la gestione dei drink;

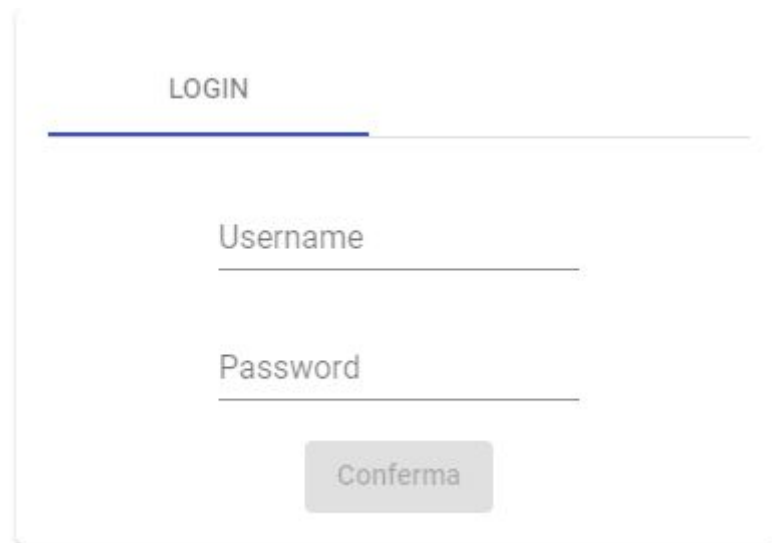
6.3 Routes

- **"/**: path del root, reindirizza automaticamente al path "/dashboard" se l'utente è autenticato, altrimenti va alla pagina di login;
- **"/login"**: path della pagina di login, se l'utente risulta autenticato, esso viene reindirizzato nella dashboard;
- **"/dashboard"**: path della pagina dashboard. Se l'utente non è autenticato, esso viene reindirizzato nella pagina di login;
- **"/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;
- **"/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.

7. Utilizzo

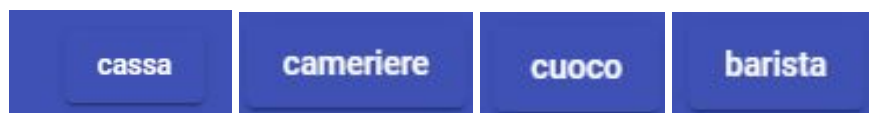
7.1 Autenticazione

- L'utente accede al sistema con le sue credenziali dalla pagina di accesso;



A login form titled "LOGIN" with a blue underline. It contains two input fields: "Username" and "Password", each with a light blue border and a small blue eye icon on the right. Below the fields is a grey button labeled "Conferma".

- É sempre possibile vedere che ruolo ha l'utente loggato guardando a destra nella navbar;



- Al termine della navigazione, l'utente può uscire dal sistema facendo clic sull'icona posta in alto a destra.



7.2 Gestione utenti

- I cassieri dalla pagina di gestione utenti possono registrare nuovi utenti andando nel tab apposito che contiene una form per l'inserimento dei dati e del ruolo del nuovo utente;

The screenshot shows a mobile application interface with three tabs: 'REGISTRAZIONE', 'UTENTI', and 'CIBI'. The 'UTENTI' tab is selected. Below the tabs is a registration form with the following fields: 'Nome' (text input), 'Cognome' (text input), 'Username' (text input), 'Ruolo' (dropdown menu), and 'Password' (text input). At the bottom of the form is a 'Conferma' button.

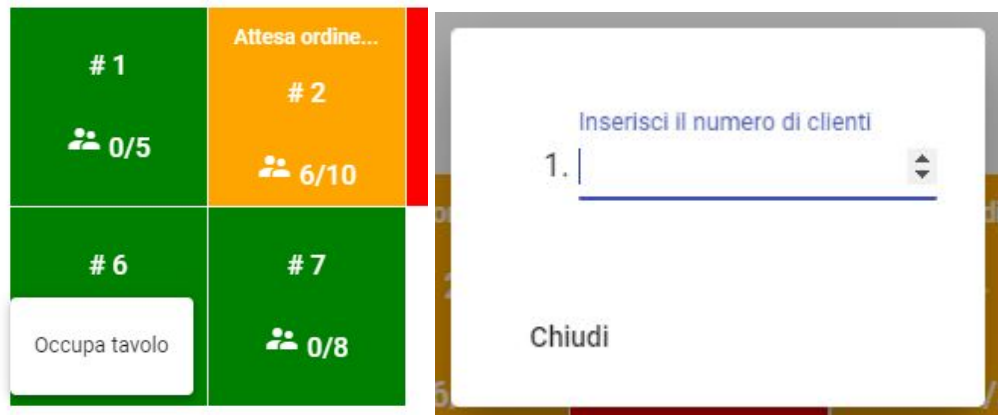
- Nella gestione utenti è possibile eliminare singoli utenti premendo il cestino di fianco a ciascun utente.

Filter

Username	Nome	Ruolo	
barista	Paolo Porcellato	barista	
cameriere	Matteo Schizzerotto	cameriere	
cuoco	Goran Gajic	cuoco	
cassa	admin admin	cassa	

7.3 Gestione ordini

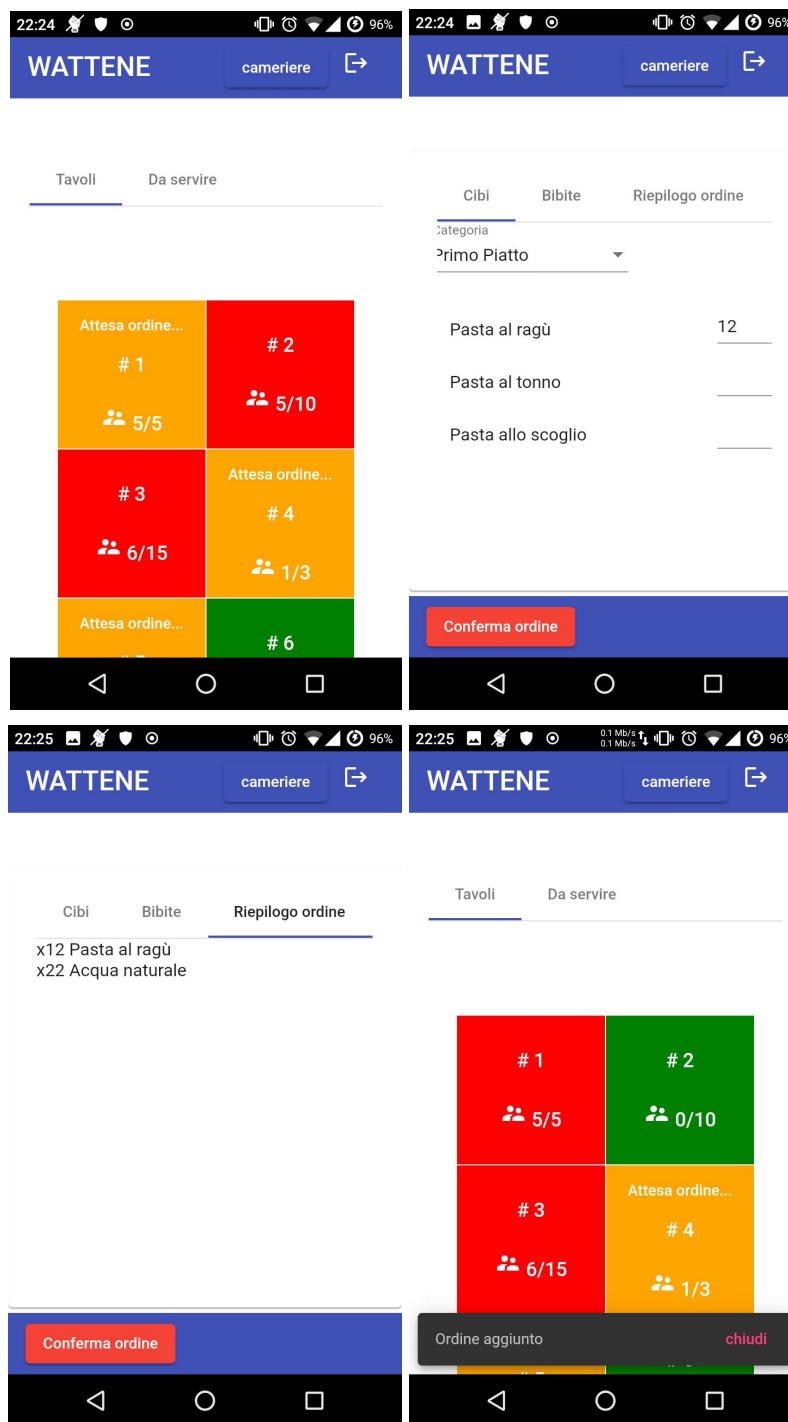
- Quando un cliente/un gruppo di clienti entra nel ristorante, un cameriere può occupare un tavolo facendo clic sul pulsante “Occupa” che viene fuori una volta cliccato sopra al tavolo;



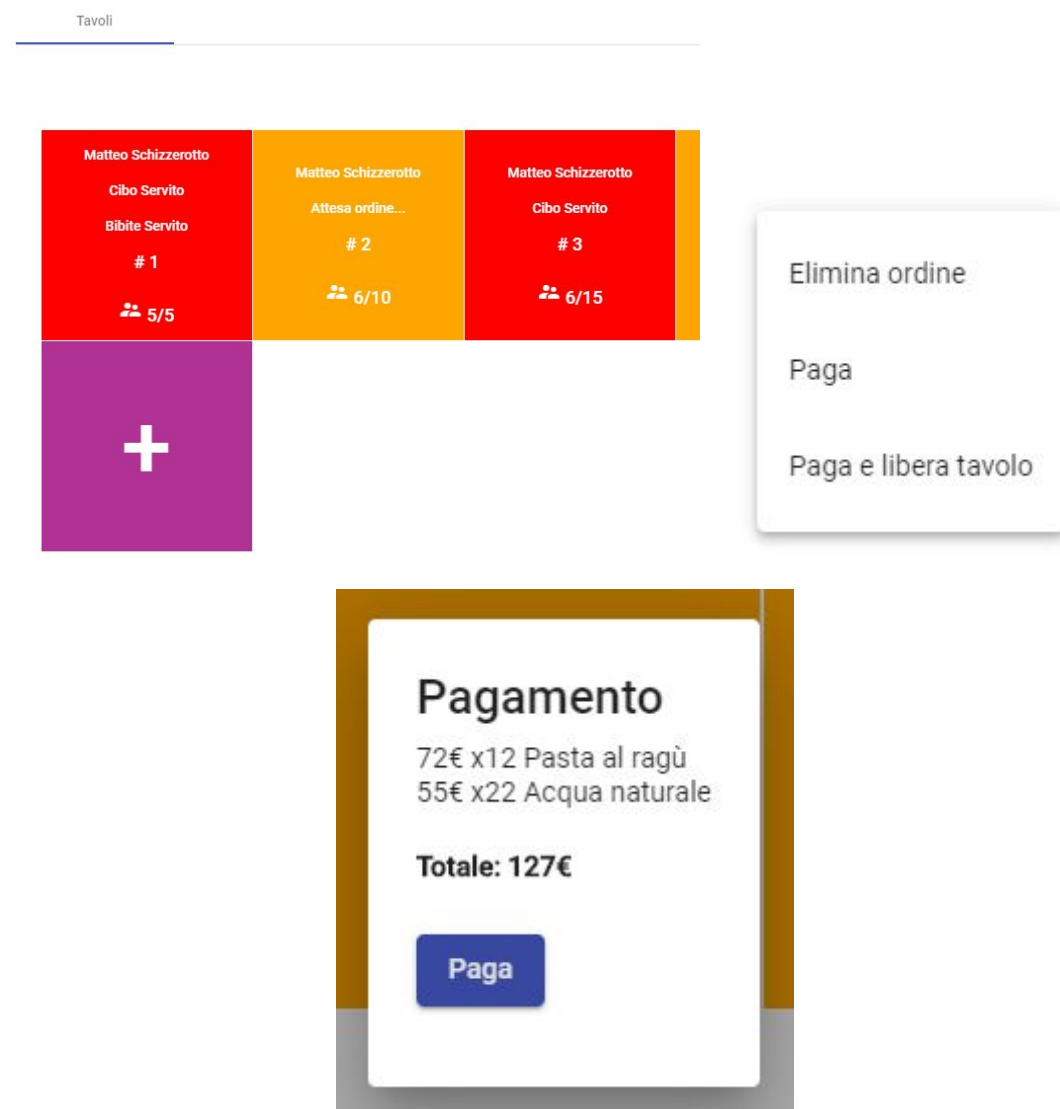
- Lo stato dell'ordine è sempre visibile dalla dashboard della cassa;



- Quando arriva il momento di prendere un ordine è sufficiente cliccare nuovamente sopra al tavolo e fare click su “Gestione ordine”;



- Il pagamento viene effettuato tramite l'interfaccia della cassa semplicemente cliccando sopra al tavolo interessato, successivamente selezionando “Paga e libera tavolo” e proseguendo con la creazione dello scontrino;



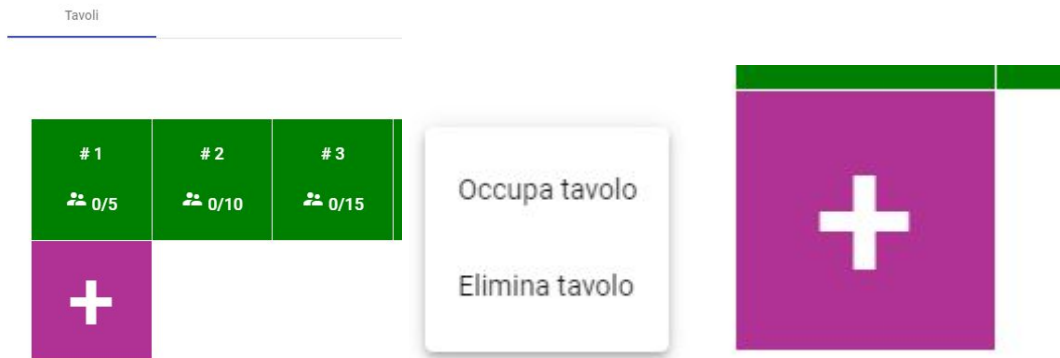
- Gestione della coda di ordini di drink o cibi.

Ordini

Pronto	Acqua frizzante x3 Tempo totale: 3 min	Sprite x2 Tempo totale: 2 min	Te alla pesca x1 Tempo totale: 1 min	
Pronto	Acqua frizzante x1 Tempo totale: 1 min	Acqua naturale x3 Tempo totale: 3 min	Coca cola x4 Tempo totale: 4 min	Fanta x5 Tempo totale: 5 min

7.4 Gestione tavoli

La cassa può eliminare o aggiungere tavoli tramite l'interfaccia della dashboard. Per eliminarne uno è sufficiente un click sopra ad esso, mentre per aggiungerne di nuovi bisogna premere su "+".



7.5 Gestione cibi e bevande

La cassa ha la possibilità di eliminare o aggiungere cibi e/o bevande dal menù in ogni momento tramite il backoffice (accessibile tramite l'icona dell'ingranaggio in alto a destra sulla navbar).

Filter

Nome	Categoria	Prezzo	Tempo	Disponibile	
Cotoletta e patatine fritte	Secondo Piatto	7.5	15	<input checked="" type="checkbox"/>	
Fagioli in salsa	Contorno	3.5	4	<input checked="" type="checkbox"/>	
Pasta allo scoglio	Primo Piatto	14	6	<input checked="" type="checkbox"/>	

Aggiungi un cibo

Clicca per visualizzare

▼

7.6 Gestione statistiche

La cassa può visualizzare varie statistiche per varie categorie di interesse.

<

Cienti

Cibi più venduti

Bibite più vendute

Camerieri

Cuochi

Baristi

>

Nome del cibo	Quantita' venduta
Fagioli in salsa	114
Pasta al ragù	12
Cotoletta e patate fritte	6
Pasta al tonno	2
Insalatona	1
Pasta allo scoglio	0
Petto di pollo ai ferri	0
Salmone ai ferri	0
Vassoio di affettati	0

<

Cienti

Cibi più venduti

Bibite più vendute

Camerieri

Cuochi

Baristi

>

Nome del cuoco	Numero piatti preparati
Goran Gajic	135

<

Cienti

Cibi più venduti

Bibite più vendute

Camerieri

Cuochi

Baristi

>

Numero clienti	Guadagni della giornata
18	709.5

8. Problemi da tenere in considerazione

- La connessione al server non è sempre sicura (viene utilizzato il protocollo HTTP nel server in locale e HTTPS nella versione che usa Heroku come host).