

TAWRESTaurant

Corso di Tecnologie e Applicazioni Web
A.A. 2018/19

Nome del gruppo

NO TAW

Membri del gruppo

- Battistich, Alvise (865154)
- Casarin, Samuele (862789)

Relazione di

Casarin, Samuele (862789)

Indice generale

1 Architettura del sistema.....	3
2 Modello dei dati.....	5
2.1 User.....	5
2.2 MenuItem.....	6
2.3 Table.....	6
2.4 Order.....	8
3 API.....	9
4 Autenticazione.....	22
5 Front-end.....	24
5.1 Components.....	24
5.2 Services.....	26
5.3 Routes.....	26
6 Workflow.....	28
6.1 Autenticazione.....	28
6.2 Gestione degli utenti.....	28
6.3 Gestione delle ordinazioni di un tavolo.....	31
7 Problemi conosciuti.....	35

1 Architettura del sistema

L'architettura del sistema è di tipo Client/Server: genericamente, un dispositivo client si connette ad un server per la fruizione di un certo servizio. Nel nostro caso, il servizio offerto consente all'utente di ottenere e modificare risorse che rappresentano lo stato dell'ambiente, ovvero il ristorante/pizzeria, per quanto riguarda la gestione delle ordinazioni. Nella pratica, il servizio è realizzato grazie all'utilizzo di tecnologie web, basate sul protocollo HTTP.

Il sistema è formato dai seguenti componenti:

- un database non relazionale orientato ai documenti;
- un servizio web di backend con API in stile REST;
- un'applicazione web, un'applicazione mobile e un'applicazione desktop (per l'interfaccia utente).

Per realizzare i vari componenti del sistema, sono state impiegate le seguenti tecnologie:

- *MongoDB*: un DBMS non relazionale orientato ai documenti (per la gestione della persistenza dei dati);
- *Node.js*: un interprete multiplattaforma 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 formato da quattro collezioni: *User*, *MenuItem*, *Table*, *Order*.

2.1 User

La collezione *User* contiene i documenti in cui sono salvate le informazioni relative agli utenti del sistema, incluse le informazioni necessarie per l'autenticazione nel sistema e varie statistiche dipendenti dal ruolo dell'utente.

Nota: assumiamo che un utente possa avere un solo ruolo.

Chiave	Tipo	Descrizione
<i>username</i>	string	Il nome univoco dell'utente. Il valore è unico tra tutti i documenti della collezione <i>User</i> .
<i>name</i>	string	Il nome dell'utente.
<i>surname</i>	string	Il cognome dell'utente.
<i>role</i>	“Waiter” “Cook” “Barman” “Cashier”	Il ruolo dell'utente (“Waiter”: cameriere; “Cook”: cuoco; “Barman”: barista; “Cashier”: cassiere).
<i>salt</i>	string	Una stringa che, insieme a <i>digest</i> , consente di mitigare gli attacchi brute-force per la decifrazione della password. Viene generato al momento della creazione/modifica della password con la seguente procedura: <ol style="list-style-type: none">1. Generare casualmente una sequenza di 16 byte;2. Codificare la sequenza di byte ottenuta come risultato del punto 1 come stringa esadecimale.
<i>digest</i>	string	Il digest della password utilizzata dall'utente per l'autenticazione nel sistema, generato con la seguente procedura: <ol style="list-style-type: none">1. Cifrare la password in chiaro con l'algoritmo HMAC basato sulla funzione hash SHA-512 utilizzando il valore di <i>salt</i> come chiave segreta;2. Codificare i byte ottenuti come risultato del punto 1 in base64.

<i>totalServedCustomers</i>	number	Il numero di clienti serviti dal cameriere. Il valore deve essere definito se <i>{role == "Waiter"}</i> . Valore iniziale: 0
<i>totalPrepareDishes</i>	number	Il numero di piatti preparati dal cuoco. Il valore deve essere definito se <i>{role == "Cook"}</i> . Valore iniziale: 0
<i>TotalPreparedBeverages</i>	number	Il numero di bevande preparati dal barista. Il valore deve essere definito se <i>{role == "Barman"}</i> Valore iniziale: 0

2.2 MenuItem

La collezione *MenuItem* contiene i documenti in cui sono salvate le informazioni relative agli elementi del menu del ristorante, ovvero i cibi e le bevande.

Chiave	Tipo	Descrizione
<i>name</i>	string	Il nome dell'elemento del menu.
<i>price</i>	number	Il prezzo dell'elemento del menu nella valuta scelta (es. in Euro).
<i>preparationTime</i>	number	Il tempo di preparazione dell'elemento del menu in minuti.
<i>kind</i>	“Food” “Beverage”	Il tipo dell'articolo di menu (“Food”: cibo, piatto; “Beverage”: bevanda).

2.3 Table

La collezione *Table* contiene i documenti in cui sono salvate le informazioni relative ai tavoli presenti nel ristorante, inclusi lo stato del tavolo e lo stato degli ordini per entrambi i tipi (ordinazioni di cibo e di bevande).

Chiave	Tipo	Descrizione
<i>number</i>	number	Il numero univoco del tavolo (es. tavolo 5). Il valore è unico tra tutti i documenti della collezione <i>Table</i> .
<i>seats</i>	number	Il numero di posti del tavolo.

<i>status</i>	“free” “not-served” “waiting” “served”	Lo stato del tavolo (“free”: tavolo libero; “not-served”: tavolo occupato, in attesa del cameriere riferito da <i>servedBy</i> per prendere le ordinazioni; “waiting”: tavolo occupato, in attesa che sia le ordinazioni di cibo che di bevande siano servite al tavolo; “served”: tavolo occupato, servito sia del cibo che delle bevande). Valore iniziale: “free”
<i>numOfCustomers</i>	number	Il numero di clienti che stanno occupando il tavolo. Il valore deve essere minore o uguale a <i>seats</i> . Il valore deve essere definito se { <i>status</i> != “free”}.
<i>servedBy</i>	ObjectId	Il riferimento al cameriere che sta servendo il tavolo. Il valore deve essere l’ID di un documento della collezione <i>User</i> dove { <i>role</i> == “Waiter”}. Il valore deve essere definito se { <i>status</i> != “free”}.
<i>occupiedAt</i>	Date	La data e l’ora in cui il tavolo è stato occupato (ovvero <i>status</i> passa da “free” a “not-served”). Il valore deve essere definito se { <i>status</i> != “free”}.
<i>ordersTakenAt</i>	Date	La data e l’ora in cui il cameriere riferito da <i>servedBy</i> ha finito di prendere le ordinazioni (ovvero <i>status</i> passa da “not-served” a “waiting”). Il valore deve essere definito se { <i>status</i> == “waiting” <i>status</i> == “served”}.
<i>foodOrdersStatus</i>	“pending” “ready” “served”	Lo stato degli ordini di cibo (“pending”: in attesa di preparazione; “ready”: tutti i piatti sono pronti per essere serviti; “served”: i piatti sono stati serviti dal cameriere riferito da <i>servedBy</i>). Il valore deve essere definito se { <i>status</i> == “waiting” <i>status</i> == “served”}.
<i>beverageOrdersStatus</i>	“pending” “ready” “served”	Lo stato degli ordini di bevande (“pending”: in attesa di preparazione; “ready”: tutte le bevande sono pronte per essere servite; “served”: le bevande sono stati servite dal cameriere riferito da <i>servedBy</i>). Il valore deve essere definito se { <i>status</i> == “waiting” <i>status</i> == “served”}.

2.4 Order

La collezione *Order* contiene i documenti in cui sono salvate le informazioni relative alle ordinazioni prese ai tavoli.

Assumiamo di eliminare dal database le ordinazioni riferite ad un dato tavolo una volta che questo viene liberato (ovvero *status* passa a “free”).

Chiave	Tipo	Descrizione
<i>table</i>	ObjectId	Il riferimento al tavolo da cui proviene l'ordinazione. Il valore deve essere l'ID di un documento della collezione <i>Table</i> .
<i>status</i>	“pending” “preparing” “ready”	Lo stato dell'ordinazione (“pending”: l'ordine è in attesa di essere preparato; “preparing”: l'ordine è in preparazione; “ready”: l'ordine è pronto per essere servito). Valore iniziale: “pending”
<i>kind</i>	“FoodOrder” “BeverageOrder”	Il tipo di ordinazione (“FoodOrder”: ordinazione di un piatto; “BeverageOrder”: ordinazione di una bevanda).
<i>food</i>	ObjectId	Il riferimento al piatto da preparare. Il valore deve essere l'ID di un documento della collezione <i>MenuItem</i> dove <i>{kind == “Food”}</i> . Il valore deve essere definito se <i>{kind == “FoodOrder”}</i> .
<i>cook</i>	ObjectId	Il riferimento al cuoco che sta preparando il cibo riferito da <i>food</i> . Il valore deve essere l'ID di un documento della collezione <i>User</i> dove <i>{role: “Cook”}</i> . Il valore deve essere definito se <i>{kind == “FoodOrder”}</i> .
<i>beverage</i>	ObjectId	Il riferimento alla bevanda da preparare. Il valore deve essere l'ID di un documento della collezione <i>MenuItem</i> dove <i>{kind == “Beverage”}</i> . Il valore deve essere definito se <i>{kind == “BeverageOrder”}</i> .
<i>barman</i>	ObjectId	Il riferimento al barista che sta preparando la bevanda riferita da <i>beverage</i> . Il valore deve essere l'ID di un documento della collezione <i>User</i> dove <i>{role: “Barman”}</i> . Il valore deve essere definito se <i>{kind == “BeverageOrder”}</i> .

3 API

In questa sezione sono descritte le API (v1.0.0) del servizio web di backend.

Nota: i dati della richiesta/risposta dei vari endpoint sono formattati in JSON, il tipo specifico del formato dei dati per ogni caso è qui rappresentato in pseudo-TypeScript.

Endpoint	/login
Metodo	POST
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ <i>token</i> : string }
Descrizione	Effettua l'autenticazione nel sistema. Le credenziali dell'utente devono essere fornite attraverso il metodo di autenticazione HTTP <i>Basic</i> . Se le credenziali sono valide, viene restituito un oggetto che contiene un nuovo JSON Web Token della durata di 12 ore di validità. Il token può essere fornito alle successive richieste che richiedono l'autenticazione nel sistema attraverso il metodo di autenticazione HTTP <i>Bearer</i> .

Endpoint	/events
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	
Descrizione	Restituisce un canale di Socket.io per l'osservazione di eventi in tempo reale. Il servizio web può emettere due tipi di eventi: “table status changed” quando lo stato di un dato tavolo è cambiato (il payload dell'evento contiene tutte le informazioni del tavolo); oppure “order status changed” quando lo stato di un dato ordine è cambiato (il payload dell'evento contiene tutte le informazioni dell'ordine).

Endpoint	/users
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>role</i> : string, <i>totalServedCustomers</i> ?: number, <i>totalPreparedDishes</i> ?: number, <i>totalPreparedBeverages</i> ?: number }]
Descrizione	Restituisce la lista di tutti gli utenti.

Endpoint	/users/byId/:id
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ <i>_id</i> : string, <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>role</i> : string, <i>totalServedCustomers</i> ?: number, <i>totalPreparedDishes</i> ?: number, <i>totalPreparedBeverages</i> ?: number }
Descrizione	Restituisce le informazioni relative all'utente con ID <i>id</i> .

Endpoint	/users/byId/:id/password
Metodo	PUT
Parametri	
Formato dei dati della richiesta	{ <i>password</i> : string }
Formato dei dati della risposta	
Descrizione	Cambia la password dell'utente con ID <i>id</i> se l'utente che ha effettuato la richiesta è un cassiere.

Endpoint	/users/byId/:id
Metodo	DELETE
Parametri	
Formato dei dati della richiesta	

Formato dei dati della risposta	
Descrizione	Elimina l'utente con ID <i>id</i> se l'utente che ha effettuato la richiesta è un cassiere.

Endpoint	/users/waiters
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>role</i> : string, <i>totalServedCustomers</i> : number }]
Descrizione	Restituisce la lista di tutti i camerieri.

Endpoint	/users/waiters
Metodo	POST
Parametri	
Formato dei dati della richiesta	{ <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>password</i> : string }
Formato dei dati della risposta	{ <i>_id</i> : string, <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>role</i> : string, <i>totalServedCustomers</i> : number }
Descrizione	Crea un nuovo cameriere se l'utente che ha effettuato la richiesta è un cassiere.

Endpoint	/users/cooks
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>role</i> : string, <i>totalPreparedDishes</i> : number }]
Descrizione	Restituisce la lista di tutti i cuochi.

Endpoint	/users/cooks
Metodo	POST

Parametri	
Formato dei dati della richiesta	{ <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>password</i> : string }
Formato dei dati della risposta	{ <i>_id</i> : string, <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>role</i> : string, <i>totalPreparedDishes</i> : number }
Descrizione	Crea un nuovo cuoco se l'utente che ha effettuato la richiesta è un cassiere.

Endpoint	/users/barmans
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>role</i> : string, <i>totalPreparedBeverages</i> : number }]
Descrizione	Restituisce la lista di tutti i baristi.

Endpoint	/users/barmans
Metodo	POST
Parametri	
Formato dei dati della richiesta	{ <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>password</i> : string }
Formato dei dati della risposta	{ <i>_id</i> : string, <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>role</i> : string, <i>totalPreparedBeverages</i> : number }
Descrizione	Crea un nuovo barista se l'utente che ha effettuato la richiesta è un cassiere.

Endpoint	/users/cashiers
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>role</i> : string }]
Descrizione	Restituisce la lista di tutti i cassieri.

Endpoint	/users/cashiers
Metodo	POST
Parametri	
Formato dei dati della richiesta	{ <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>password</i> : string }
Formato dei dati della risposta	{ <i>_id</i> : string, <i>username</i> : string, <i>name</i> : string, <i>surname</i> : string, <i>role</i> : string }
Descrizione	Crea un nuovo cassiere se l'utente che ha effettuato la richiesta è un cassiere.

Endpoint	/menu
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>name</i> : string, <i>price</i> : string, <i>preparationTime</i> : number, <i>kind</i> : string }]
Descrizione	Restituisce la lista di tutti i cibi e le bevande presenti nel menu.

Endpoint	/menu/byId/:id
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ <i>_id</i> : string, <i>name</i> : string, <i>price</i> : string, <i>preparationTime</i> : number, <i>kind</i> : string }
Descrizione	Restituisce il cibo/la bevanda presente nel menu con ID <i>id</i> .

Endpoint	/menu/foods
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>name</i> : string, <i>price</i> : string, <i>preparationTime</i> : number, <i>kind</i> : string }]
Descrizione	Restituisce la lista di tutti i cibi presenti nel menu.

Endpoint	/menu/beverages
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>name</i> : string, <i>price</i> : string, <i>preparationTime</i> : number, <i>kind</i> : string }]
Descrizione	Restituisce la lista di tutte le bevande presenti nel menu.

Endpoint	/tables
Metodo	GET
Parametri	<i>seats</i> : filtro di ricerca numerico per numero di posti (limite inferiore). <i>status</i> : filtro di ricerca per stato del tavolo. <i>foodOrdersStatus</i> : filtro di ricerca per stato delle ordinazioni di cibo. <i>beverageOrdersStatus</i> : filtro di ricerca per stato delle ordinazioni di bevande.
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>number</i> : number, <i>seats</i> : number, <i>status</i> : string, <i>numOfCustomers</i> : number, <i>servedBy</i> : string, <i>occupiedAt</i> : DateString, <i>ordersTakenAt</i> : DateString, <i>foodOrdersStatus</i> : string, <i>beverageOrdersStatus</i> : string }]
Descrizione	Restituisce la lista di tutti i tavoli con i filtri applicati. <i>servedBy</i> è l'ID del cameriere che sta servendo il tavolo.

Endpoint	/users/waiters/byId/:id/tables
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>number</i> : number, <i>seats</i> : number, <i>status</i> : string, <i>numOfCustomers</i> : number, <i>servedBy</i> : string, <i>occupiedAt</i> : DateString, <i>ordersTakenAt</i> : DateString, <i>foodOrdersStatus</i> : string, <i>beverageOrdersStatus</i> : string}]

	string }]
Descrizione	Restituisce la lista di tutti i tavoli serviti dal cameriere con ID <i>id</i> . <i>servedBy</i> è l'ID del cameriere che sta servendo il tavolo.

Endpoint	/tables/byId/:id
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ _id: string, number: number, seats: number, status: string, numOfCustomers: number, servedBy: string, occupiedAt: DateString, ordersTakenAt: DateString, foodOrdersStatus: string, beverageOrdersStatus: string }
Descrizione	Restituisce tutte le informazioni relative al tavolo con ID <i>id</i> . <i>servedBy</i> è l'ID del cameriere che sta servendo il tavolo.

Endpoint	/tables/byId/:id
Metodo	PUT
Parametri	<i>action</i> (obbligatorio): l'azione di modifica dello stato del tavolo con ID <i>id</i> , può essere “occupy” (occupa il tavolo se l'utente che ha effettuato la richiesta è un cameriere e il tavolo è libero) oppure “free” (libera il tavolo se l'utente che ha effettuato la richiesta è un cassiere e il tavolo non è libero).
Formato dei dati della richiesta	
Formato dei dati della risposta	
Descrizione	Esegue l'azione specificata sul tavolo con ID <i>id</i> .

Endpoint	/tables/byId/:id/orders
Metodo	GET
Parametri	<i>status</i> : filtro di ricerca per lo stato delle ordinazioni.
Formato dei dati	

della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>table</i> : string, <i>status</i> : string, <i>kind</i> : string, <i>food</i> ?: { <i>_id</i> : string, <i>name</i> : string, <i>price</i> : number, <i>preparationTime</i> : number, <i>kind</i> : string }, <i>cook</i> ?: string, <i>beverage</i> ?: { <i>_id</i> : string, <i>name</i> : string, <i>price</i> : number, <i>preparationTime</i> : number, <i>kind</i> : string }, <i>barman</i> ?: string }]
Descrizione	<p>Restituisce la lista di tutti gli ordini del tavolo con ID <i>id</i>.</p> <p><i>food</i> è il cibo da preparare.</p> <p><i>cook</i> è l'ID del cuoco a cui è assegnata la preparazione del piatto.</p> <p><i>food</i> e <i>cook</i> sono definiti se <i>{kind == "FoodOrder"}</i>.</p> <p><i>beverage</i> è la bevanda da preparare.</p> <p><i>barman</i> è l'ID del barista a cui è assegnata la preparazione della bevanda.</p> <p><i>beverage</i> e <i>barman</i> sono definiti se <i>{kind == "BeverageOrder"}</i>.</p>

Endpoint	/tables/byId/:idT/orders/byId/:idO
Metodo	GET
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	{ <i>_id</i> : string, <i>table</i> : string, <i>status</i> : string, <i>kind</i> : string, <i>food</i> ?: { <i>_id</i> : string, <i>name</i> : string, <i>price</i> : number, <i>preparationTime</i> : number, <i>kind</i> : string }, <i>cook</i> ?: string, <i>beverage</i> ?: { <i>_id</i> : string, <i>name</i> : string, <i>price</i> : number, <i>preparationTime</i> : number, <i>kind</i> : string }, <i>barman</i> ?: string }
Descrizione	<p>Restituisce tutte le informazioni relative all'ordine con id <i>idO</i> del tavolo con ID <i>idT</i>.</p> <p><i>food</i> è il cibo da preparare.</p> <p><i>cook</i> è l'ID del cuoco a cui è assegnata la preparazione del piatto.</p> <p><i>food</i> e <i>cook</i> sono definiti se <i>{kind == "FoodOrder"}</i>.</p> <p><i>beverage</i> è la bevanda da preparare.</p> <p><i>barman</i> è l'ID del barista a cui è assegnata la preparazione della bevanda.</p> <p><i>beverage</i> e <i>barman</i> sono definiti se <i>{kind == "BeverageOrder"}</i>.</p>

Endpoint	/tables/byId/:id/orders/foodOrders
Metodo	GET

Parametri	<i>status</i> : filtro di ricerca per lo stato delle ordinazioni.
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>table</i> : string, <i>status</i> : string, <i>kind</i> : string, <i>food</i> : { <i>_id</i> : string, <i>name</i> : string, <i>price</i> : number, <i>preparationTime</i> : number, <i>kind</i> : string }, <i>cook</i> : string }]
Descrizione	Restituisce la lista di tutti gli ordini di cibo del tavolo con ID <i>id</i> . <i>food</i> è il cibo da preparare. <i>cook</i> è l'ID del cuoco a cui è assegnata la preparazione del piatto.

Endpoint	/users/cooks/byId/: <i>id</i> /orders
Metodo	GET
Parametri	<i>status</i> : filtro di ricerca per lo stato delle ordinazioni.
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>table</i> : string, <i>status</i> : string, <i>kind</i> : string, <i>food</i> : { <i>_id</i> : string, <i>name</i> : string, <i>price</i> : number, <i>preparationTime</i> : number, <i>kind</i> : string }, <i>cook</i> : string }]
Descrizione	Restituisce la lista di tutti gli ordini assegnati al cuoco con ID <i>id</i> . <i>food</i> è il cibo da preparare. <i>cook</i> è l'ID del cuoco a cui è assegnata la preparazione del piatto.

Endpoint	/tables/byId/: <i>id</i> /orders/beverageOrders
Metodo	GET
Parametri	<i>status</i> : filtro di ricerca per lo stato delle ordinazioni.
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>table</i> : string, <i>status</i> : string, <i>kind</i> : string, <i>beverage</i> : { <i>_id</i> : string, <i>name</i> : string, <i>price</i> : number, <i>preparationTime</i> : number, <i>kind</i> : string }, <i>barman</i> : string }]
Descrizione	Restituisce la lista di tutti gli ordini di bevande del tavolo con ID <i>id</i> . <i>beverage</i> è la bevanda da preparare. <i>barman</i> è l'ID del barista a cui è assegnata la preparazione della bevanda.

Endpoint	/users/barmans/byId/:id/orders
Metodo	GET
Parametri	<i>status</i> : filtro di ricerca per lo stato delle ordinazioni.
Formato dei dati della richiesta	
Formato dei dati della risposta	[{ <i>_id</i> : string, <i>table</i> : string, <i>status</i> : string, <i>kind</i> : string, <i>beverage</i> : { <i>_id</i> : string, <i>name</i> : string, <i>price</i> : number, <i>preparationTime</i> : number, <i>kind</i> : string }, <i>barman</i> : string }]
Descrizione	Restituisce la lista di tutti gli ordini di bevande assegnati al barista con ID <i>id</i> . <i>beverage</i> è la bevanda da preparare. <i>barman</i> è l'ID del barista a cui è assegnata la preparazione della bevanda.

Endpoint	/tables/byId/:id/orders/foodOrders
Metodo	POST
Parametri	<i>status</i> : filtro di ricerca per lo stato delle ordinazioni.
Formato dei dati della richiesta	{ <i>food</i> : string }
Formato dei dati della risposta	{ <i>_id</i> : string, <i>table</i> : string, <i>status</i> : string, <i>kind</i> : string, <i>food</i> : { <i>_id</i> : string, <i>name</i> : string, <i>price</i> : number, <i>preparationTime</i> : number, <i>kind</i> : string }, <i>cook</i> : string }
Descrizione	Crea un ordine di cibo per il tavolo con ID <i>id</i> se l'utente che ha effettuato la richiesta è lo stesso cameriere che sta servendo il tavolo e <i>status</i> del tavolo è “not-served”. <i>food</i> nei dati della richiesta è l'ID del cibo da preparare, mentre nei dati della risposta è il cibo da preparare.

Endpoint	/tables/byId/:id/orders/beverageOrders
Metodo	POST
Parametri	
Formato dei dati della richiesta	{ <i>beverage</i> : string }
Formato dei dati della risposta	{ <i>_id</i> : string, <i>table</i> : string, <i>status</i> : string, <i>kind</i> : string, <i>beverage</i> : { <i>_id</i> : string, <i>name</i> : string, <i>price</i> : number,

	<i>preparationTime: number, kind: string }, barman: string }</i>
Descrizione	Crea un ordine di bevanda per il tavolo con ID <i>id</i> se l'utente che ha effettuato la richiesta è lo stesso cameriere che sta servendo il tavolo e <i>status</i> del tavolo è “not-served”. <i>beverage</i> nei dati della richiesta è l'ID della bevanda da preparare, mentre nei dati della risposta è la bevanda da preparare.

Endpoint	/tables/byId/:idT/orders/byId/:idO
Metodo	DELETE
Parametri	
Formato dei dati della richiesta	
Formato dei dati della risposta	
Descrizione	Elimina l'ordine con ID <i>idO</i> del tavolo con ID <i>idT</i> se l'utente che ha effettuato la richiesta è lo stesso cameriere che sta servendo il tavolo e <i>status</i> del tavolo è “not-served”.

Endpoint	/tables/byId/:id/orders
Metodo	PUT
Parametri	<i>action</i> (obbligatorio): azione di aggiornamento dello stato del tavolo con ID <i>id</i> , può solo essere “commit” (consegna da parte del cameriere che sta servendo il tavolo delle ordinazioni del tavolo in cucina/bar se <i>status</i> del tavolo è “not-served” ed esiste almeno un ordine di cibo o bevanda).
Formato dei dati della richiesta	
Formato dei dati della risposta	
Descrizione	Esegue l'azione specificata sul tavolo con ID <i>id</i> .

Endpoint	/tables/byId/:id/orders/foodOrders
Metodo	PUT
Parametri	<i>action</i> (obbligatorio): azione di aggiornamento dello stato del tavolo con ID <i>id</i> , può solo essere “serve”

	(notifica da parte del cameriere che sta servendo il tavolo che i piatti sono stati serviti al tavolo se <i>foodOrdersStatus</i> è “ready”).
Formato dei dati della richiesta	
Formato dei dati della risposta	
Descrizione	Esegue l’azione specificata sul tavolo con ID <i>id</i> .

Endpoint	/tables/byId/:id/orders/beverageOrders
Metodo	PUT
Parametri	<i>action</i> (obbligatorio): azione di aggiornamento dello stato del tavolo con ID <i>id</i> , può solo essere “serve” (notifica da parte del cameriere che sta servendo il tavolo che le bevande sono stati servite al tavolo se <i>beverageOrdersStatus</i> è “ready”).
Formato dei dati della richiesta	
Formato dei dati della risposta	
Descrizione	Esegue l’azione specificata sul tavolo con ID <i>id</i> .

Endpoint	/tables/byId/:idT/orders/foodOrders/byId/:idO
Metodo	PUT
Parametri	<i>action</i> (obbligatorio): azione di aggiornamento dello stato dell’ordine di cibo con ID <i>idO</i> del tavolo con ID <i>idT</i> , può essere “assign” (assegnazione dell’ordine per la preparazione se l’utente che ha effettuato la richiesta è un cuoco, <i>status</i> del tavolo è “waiting” e <i>status</i> dell’ordine è “pending”) oppure “notify” (notifica da parte del cuoco a cui è assegnato l’ordine che il piatto è pronto per essere servito se <i>status</i> dell’ordine è “preparing”).
Formato dei dati della richiesta	
Formato dei dati della risposta	
Descrizione	Esegue l’azione specificata sul tavolo con ID <i>id</i> .

Endpoint	/tables/byId/:idT/orders/beverageOrders/byId/:idO
-----------------	---

Metodo	PUT
Parametri	<i>action</i> (obbligatorio): azione di aggiornamento dello stato dell'ordine di bevanda con ID <i>idO</i> del tavolo con ID <i>idT</i> , può essere “assign” (assegnazione dell'ordine per la preparazione se l'utente che ha effettuato la richiesta è un barista, <i>status</i> del tavolo è “waiting” e <i>status</i> dell'ordine è “pending”) oppure “notify” (notifica da parte del barista a cui è assegnato l'ordine che la bevanda è pronta per essere servita se <i>status</i> dell'ordine è “preparing”).
Formato dei dati della richiesta	
Formato dei dati della risposta	
Descrizione	Esegue l'azione specificata sul tavolo con ID <i>id</i> .

4 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) codificate in modo blando in base64. Questo metodo viene impiegato in questo progetto solo al momento dell'accesso al sistema.

Il secondo consiste nell'invio di una qualche 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 trasmesso attraverso una connessione sicura (con il protocollo HTTPS), che il progetto attualmente non supporta.

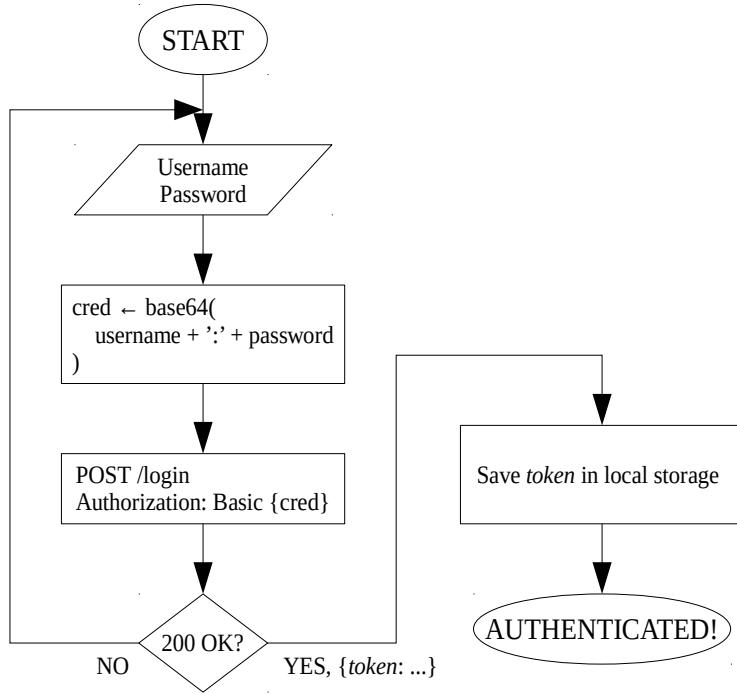


Illustrazione 1: Diagramma di flusso del processo di autenticazione

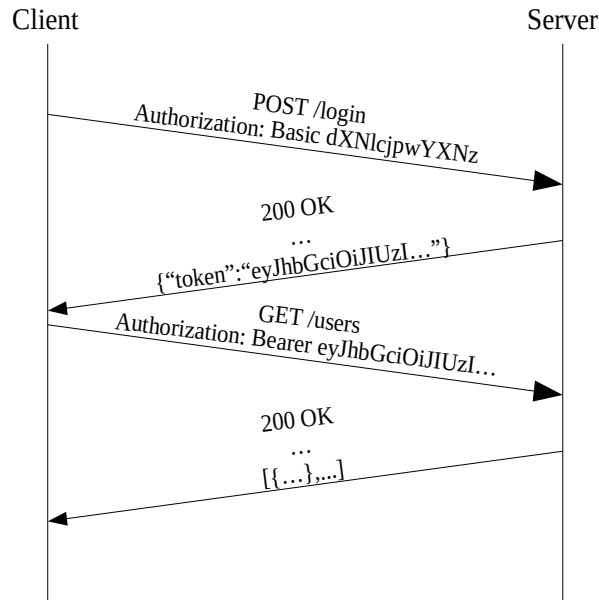


Illustrazione 2: Diagramma di sequenza del processo di autenticazione e richiesta di una risorsa protetta

5 Front-end

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.

5.1 Components

- *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*: la barra di navigazione, con i collegamenti delle pagine visitabili dall'utente e il profilo dell'utente;
- *navs-pages*: il componente della barra di navigazione specifico per la visualizzazione dei collegamenti delle pagine visitabili dall'utente;
- *nav-user-profile*: il componente della barra di navigazione specifico per la visualizzazione del profilo dell'utente;
- *login*: la pagina di accesso, che contiene un modulo per l'immissione delle credenziali dell'utente;
- *info*: la pagina con le informazioni relative all'applicazione e al progetto;
- *user-content*: un componente per la visualizzazione di un oggetto che rappresenta un utente;
- *menu-item-content*: un componente per la visualizzazione di un oggetto che rappresenta un cibo/una bevanda del menu del ristorante;
- *table-content*: un componente per la visualizzazione di un oggetto che rappresenta un tavolo;
- *order-content*: un componente per la visualizzazione di un oggetto che rappresenta un'ordinazione;

- *live-tables*: un componente per la sincronizzazione in tempo reale con il backend di una lista di tavoli, con filtraggio e ordinamento selettivi;
- *live-orders*: un componente per la sincronizzazione in tempo reale con il backend di una lista di ordinazioni, con filtraggio e ordinamento selettivi;
- *waiter-tables-page*: la pagina privata per i camerieri, che permette di visualizzare in tempo reale i tavoli in attesa di essere serviti;
- *occupy-table-modal-content*: il contenuto di una finestra modale, raggiungibile da *waiter-tables-page*, per la ricerca di tavoli liberi in base al numero di clienti e occupazione di un tavolo;
- *take-orders-modal-content*: il contenuto di una finestra modale, raggiungibile da *waiter-tables-page* e contestuale ad ogni tavolo servito dal cameriere, per prendere le ordinazioni del tavolo;
- *kitchen-page*: la pagina privata per i cuochi, permette di visualizzare in tempo reale la coda delle ordinazioni con logica FIFO, assegnare l'ordine con massimo tempo di preparazione tra quelli in attesa di essere preparati dal tavolo in cima alla coda, visualizzazione dell'ordine in preparazione e notifica da parte del cuoco della sua avvenuta preparazione; lo stesso componente è utilizzato per la pagina privata per i baristi;
- *tables-page*: la pagina privata per i cassieri, consente di visualizzare lo stato globale dei tavoli e delle ordinazioni;
- *bill-modal-content*: il contenuto di una finestra modale, raggiungibile da *tables-page* e contestuale ad ogni tavolo, per visualizzare il conto del tavolo con la lista delle ordinazioni del tavolo, il prezzo totale e un bottone per liberare il tavolo;
- *users-page*: la pagina privata per il cassiere, permette di visualizzare la lista degli utenti del sistema e le statistiche ad essi correlate e di effettuare operazioni su ognuno di essi;
- *create-user-modal-content*: il contenuto di una finestra modale, raggiungibile da *users-page*, permette la creazione di un nuovo utente;
- *change-password-modal-content*: il contenuto di una finestra modale, raggiungibile da *users-page* e contestuale ad ogni utente, permette il cambio della password dell'utente.

5.2 Services

- *root-guard*: guardia del router posta sulla route radice “/” che consente il reindirizzamento alla pagina principale dell’utente corrente;
- *auth*: servizio di gestione locale del JWT e delle informazioni relative all’utente autenticato;
- *no-auth-guard*: guardia del router che attiva la route radice “/” se l’utente è autenticato, provocando così il reindirizzamento alla pagina corretta;
- *auth-guard*: guardia del router che attiva la route radice “/” se l’utente non è autenticato, provocando così il reindirizzamento alla pagina corretta;
- *login*: servizio che presenta i metodi per l’autenticazione nel sistema da parte dell’utente;
- *users*: servizio che presenta i metodi per la gestione degli utenti;
- *menu-items*: servizio che presenta i metodi per la gestione del menu del ristorante, ovvero dell’elenco di cibi/bevande;
- *tables*: servizio che presenta i metodi per la gestione dei tavoli;
- *orders*: servizio che presenta i metodi per la gestione delle ordinazioni;
- *events*: servizio che presenta i metodi per la gestione degli eventi real-time di Socket.io.

5.3 Routes

- */* → attivazione controllata dal servizio *root-guard*;
- */login* → componente *login*, attivazione controllata dal servizio *no-auth-guard*;
- */info* → componente *info*;

- */waiter/tables* → componente *waiter-tables-page*, attivazione controllata dal servizio *auth-guard-service*, accessibile solamente da camerieri;
- */cook/kitchen* → componente *kitchen-page*, attivazione controllata dal servizio *auth-guard-service*, accessibile solamente da cuochi, pagina per gestione delle ordinazioni di cibo;
- */barman/bar* → componente *kitchen-page*, attivazione controllata dal servizio *auth-guard-service*, accessibile solamente da baristi, pagina per gestione delle ordinazioni di bevande;
- */cashier/tables* → componente *tables-page*, attivazione controllata dal servizio *auth-guard-service*, accessibile solamente da cassieri;
- */cashier/users* → componente *users-page*, attivazione controllata dal servizio *auth-guard-service*, accessibile solamente da cassieri;
- (*altro*) → reindirizzamento a “/”.

6 Workflow

6.1 Autenticazione

1. L'**utente** accede al sistema con le sue credenziali dalla pagina di accesso:

The screenshot shows a login form titled "Accedi". It has two input fields: "Nome utente" containing "cashier1" and "Password" containing "*****". Below the password field is a blue "Accedi" button.

2. Al termine della navigazione, l'**utente** può uscire dal sistema facendo clic sull'icona del profilo posta in alto a destra e su "Esci":

The screenshot shows a dashboard with a header bar "TAWRESTaurant" and navigation links "Tavoli" and "Utenti". On the right, there's a user profile icon with the name "cashier1", the role "Paperon De' Paperoni", and the status "Ruolo: Cashier". Below the profile is a blue "Esci" button. The main content area displays four tables (Tavoli 1-4) with their status as "free" and the number of available seats.

Tavolo	Stato	Numero di posti
Tavolo 1	free	4
Tavolo 2	free	6
Tavolo 3	free	2
Tavolo 4	free	

6.2 Gestione degli utenti

1. Un **cassiere** può accedere, facendo clic su "Utente" presente sulla barra di navigazione, alla pagina di gestione degli utenti, da

cui può visualizzare la lista di tutti gli utenti registrati nel sistema e le statistiche per ciascun utente:

The screenshot shows a user interface for managing restaurant staff. At the top, there's a blue header bar with the text "TAWRESTaurant" and "Tavoli Utenti". Below the header, there's a green button labeled "Crea utente" (Create user). The main area displays four user profiles in cards:

- waiter1**: Qui Non c'è, Ruolo: Waiter. Statistics: Numero di clienti serviti: 0.
- cook1**: Paolino Paperino, Ruolo: Cook. Statistics: Numero di piatti preparati: 0.
- waiter2**: Qua Nephew, Ruolo: Waiter. Statistics: Numero di clienti serviti: 0.
- cook2**: Nonna Panera, Ruolo: Cook. Statistics: Numero di piatti preparati: 0.

Each card has a small gear icon with a dropdown arrow at the top right.

2. Facendo clic su “Crea utente”, è possibile creare un nuovo utente compilando il modulo presente nella finestra modale che compare e facendo clic su “Salva”:

The screenshot shows the "Crea utente" (Create user) modal dialog. It contains fields for entering new user information:

- Nome utente: waiter10
- Nome: Pluto
- Cognome: Cane
- Password: *****
- Ruolo: Waiter

At the bottom right of the dialog is a blue "Salva" (Save) button.

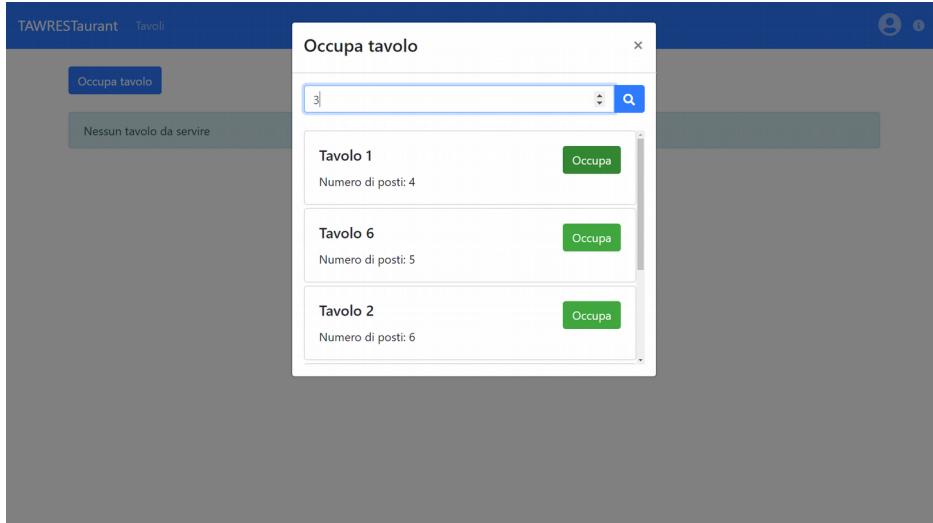
3. Facendo clic sull'icona posta affianco ad un utente, è possibile scegliere un'operazione da effettuare sull'utente, tra cui cambiare la password dell'utente ed eliminare l'utente:

4. Facendo clic su “Cambia password”, è possibile cambiare la password dell’utente immettendo la nuova password nel modulo presente nella finestra modale che compare e facendo clic su “Salva”:

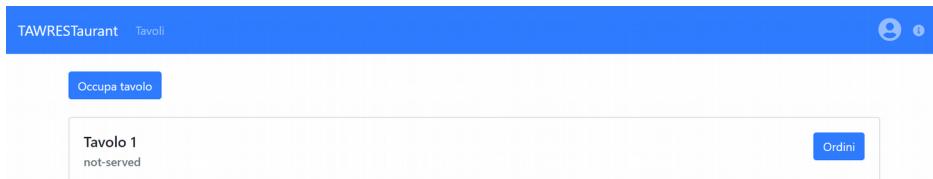
5. Facendo clic su “Elimina”, è possibile eliminare l’utente previa conferma:

6.3 Gestione delle ordinazioni di un tavolo

- Quando un cliente/un gruppo di clienti entra nel ristorante, un **cameriere** naviga alla pagina “Tavoli”, cerca un tavolo da occupare facendo clic su “Occupava tavolo” e immettendo nella barra di ricerca presente nella finestra modale che compare il numero dei clienti e infine, se trova un tavolo libero, può occuparlo facendo clic sul pulsante “Occupava” posto affianco al tavolo:

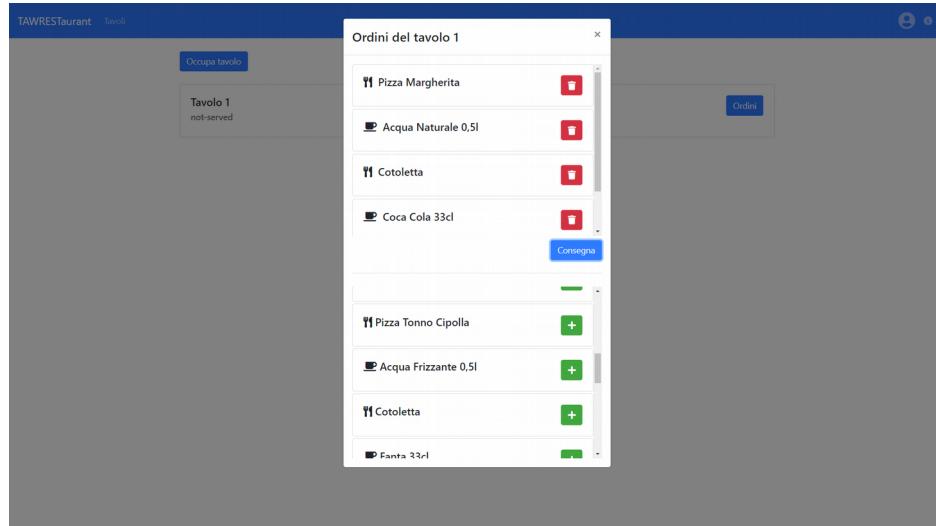


- Dalla pagina “Tavoli”, il **cameriere** può visualizzare la lista di tutti i tavoli che sta servendo al momento:

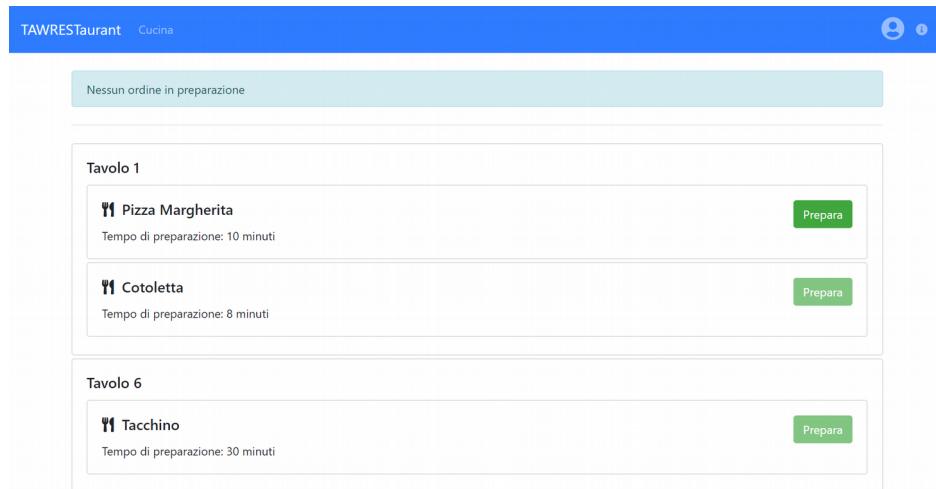


- Il **cameriere** può prendere gli ordini di un tavolo non ancora servito facendo clic sul pulsante “Ordini” posto affianco al tavolo; nella finestra modale che compare sono presenti due sezioni: la sezione in alto comprende la lista delle ordinazioni, con la possibilità di rimuoverle singolarmente facendo clic sull’icona , mentre la sezione in basso presenta il menu del ristorante, con la possibilità di

aggiungere un'ordinazione di tale piatto/bevande facendo clic sull'icona **+**; al termine delle ordinazioni, facendo clic su “Consegna”, il **cameriere**, previa conferma, può consegnare le ordinazioni in cucina/bar per la preparazione:



- Una volta consegnate le ordinazioni in cucina/bar, queste vengono aggiunte nella rispettiva coda di preparazione; facendo clic sul pulsante “Prepara” posto affianco all’ordinazione, questa viene assegnata al **cuoco/barman** corrente:



- Il **cuoco/barman** corrente può visualizzare l’ordine che sta preparando in alto; al termine della preparazione del piatto/della bevanda, può notificare l’evento facendo clic su “Pronto”:

The screenshot shows the 'Cucina' section of the TAWRESTaurant app. It displays three orders:

- Pizza Margherita**: Tempo di preparazione: 10 minuti. A green 'Pronto' button is visible.
- Cotoletta**: Tempo di preparazione: 8 minuti. A green 'Prepara' button is visible.
- Tacchino**: Tempo di preparazione: 30 minuti. A green 'Prepara' button is visible.

- Quando tutte le ordinazioni del tavolo sono pronte, il **cameriere** viene notificato dell'evento e serve i piatti/le bevande al tavolo; una volta serviti, può notificare l'evento facendo clic su “Piatti serviti”/“Bevande servite”:

The screenshot shows the 'Tavoli' section of the TAWRESTaurant app. It displays a table status update:

- Tavolo 1**: waiting
- Piatti: ready**
- Bevande: served**
- A blue 'Piatti serviti' button is visible.

- Parallelamente, il **cassiere** può visualizzare in tempo reale lo stato di tutti i tavoli e di tutte le ordinazioni dei tavoli:

TAVOLE

Tavolo 1 served Numero di posti: 4	Conto
Tavolo 2 free Numero di posti: 6	
Tavolo 3 free Numero di posti: 2	
Tavolo 4 free Numero di posti: 10	
Tavolo 5 free Numero di posti: 8	

8. Una volta serviti e terminato il pasto, i clienti del tavolo si rivolgono ad un **cassiere** per pagare il conto; il **cassiere** cerca il tavolo in questione nella lista di tavoli e, facendo clic su “Conto”, visualizza nella finestra modale che compare lo scontrino con la lista delle ordinazioni con il prezzo per ognuna di esse e il totale; dopo il pagamento, il **cassiere** libera il tavolo facendo clic su “Fine”:

CONTI

Conto del tavolo 1

🍕 Pizza Margherita	€ 5
☕ Acqua Naturale 0,5l	€ 1.5
🥩 Cotoletta	€ 8
Totale: € 17.5	

Fine

7 Problemi conosciuti

- La connessione al server non è sicura (viene utilizzato il protocollo HTTP);
- Gli endpoint delle API potrebbero essere potenzialmente soggetti ad attacchi CSRF (Cross Site Request Forgery);
- La validazione dei dati inviati al server è blanda e facilmente violabile;
- L'applicazione mobile per Android creata con Cordova non permette la connessione al server all'indirizzo <http://10.0.2.2:3201> dall'emulatore (dove 10.0.2.2 è l'indirizzo IP del dispositivo host e 3201 la porta scelta per il *socket* del server), nonostante le politiche di *origin* impostate dovrebbero permettere la connessione per ogni indirizzo.