



Università
Ca'Foscari
Venezia

Università Ca' Foscari
Laurea Triennale in Informatica

Corso di Tecnologie e Applicazioni Web [CT0142]
A.A. 2018-19

Relazione progetto d'esame

Pizzeria

MARTIGNON Nicolas Pietro (870034)

Sommario

Architettura del sistema 3

Modello dei dati 5

Lista endpoints 9

Autenticazione degli utenti..... 12

Workflow dell'applicazione 15

Architettura del sistema

L'architettura che sta alla base del sistema è composta da:

- Un web service di backend con API in stile REST, implementato in Typescript su ambiente Node.js.
- Un DBMS MongoDB per gestire la persistenza dei dati
- Una web application di front-end in stile SPA realizzata con il framework Angular, corredata di applicazione mobile android realizzata con Apache Cordova e applicazione desktop Windows realizzata con Electron.

L'intero sistema è stato scritto con il linguaggio di programmazione Typescript: Super-set di JavaScript che ne estende la sintassi, aggiungendo tra le molte cose:

- Firma dei metodi
- Classi
- Interfacce
- Tipi di dato (opzionali)
- Tipo Enum
- Mixin tra classi

Per soddisfare le features richieste sono stati utilizzati dei moduli node (Node Modules) che altro non sono che delle librerie javascript installabili attraverso npm (Node Package Manager). Questi moduli sono stati resi compatibili con il linguaggio typescript importando i TypeScript type definitions per ogni modulo.

Qui di seguito sono elencati tutti i moduli utilizzati insieme alle funzionalità che aggiungono al sistema.

Node modules Server:

- **Colors:** Permette la personalizzazione dell'output visivo sulla console javascript, consente quindi di cambiare lo stile il colore e il colore di background del testo.
- **Mongoose:** E' uno strumento di modellazione di oggetti MongoDB progettato per funzionare in un ambiente asincrono, permette di definire lo schema dei documenti attraverso oggetti Javascript.
- **Jsonwebtoken:** Utilizzato per la generazione dei token.
- **Socket.io:** Modulo utilizzato per effettuare comunicazioni bidirezionali realtime attraverso web-socket.
- **Express:** E' un middleware che permette di gestire il routing tra i vari endpoint e API, inoltre permette l'utilizzo di moltissimi HTTP utility/moduli che leggono/modificano in sequenza gli oggetti request e response.

Utility principali utilizzate nel progetto:

- **Body-parser:** Effettua il parsing del body della richiesta http, il risultato è disponibile in req.body

- **Passport:** Modulo che gestisce l'autenticazione delle richieste
- **Passport-HTTP:** Permette di autenticare richieste http usando basic authentication e digest.
- **Express-jwt:** Permette l'autenticazione delle richieste http usando JWT tokens, se il token è valido, a req.user sarà assegnato l'oggetto JSON contenuto nel JWT.

Node modules Angular Client:

- **Bootstrap:** Framework grafico per la creazione di siti responsivi e applicazioni per il Web, basati su HTML e CSS.
- **Socket.io-client:** Modulo utilizzato per ricevere comunicazioni realtime attraverso web-socket.
- **Jquery:** Libreria javascript molto diffusa per la manipolazione del DOM, gestione eventi e in generale la gestione di un sito web. Con il tempo è diventata per l'appunto uno standard de facto.
- **Jwt-decode:** Modulo che permette la decrittazione Base64Url dei JWT.

Modello dei dati

Per la gestione della persistenza dei dati si è utilizzato MongoDB un database non relazionale che organizza i dati attraverso dei documenti (oggetti JSON), raggruppati in collezioni.

Collezioni utilizzate con i loro relativi schemi:

- **User**

-username	string
-cognome	string
-nome	string
-ruolo	string
-salt	string
-digest	string

Collezione che contiene tutti i dipendenti della pizzeria, in particolare salt e digest servono per settare e validare la password in modo sicuro.

- **Elenco_tavoli**

-numero	number
-posti	number

Il campo numero contiene il numero del tavolo stesso, il campo posti invece identifica il numero massimo di persone che il tavolo può accogliere.

- **Listino_pizze**

-nome	string
-prezzo	number
-descrizione	string
-tempo_di_preparazione	number

- **Listino_bibite**

-nome	string
-prezzo	number
-descrizione	string

Si è deciso per comodità di dividere i prodotti acquistabili, in pizze e bibite che differiscono solo per il fatto che le pizze hanno il campo tempo_di_preparazione, tale campo risulta necessario per ordinare le pizze da preparare. In questo modo nella coda di preparazione

del pizzaiolo verranno mostrate le pizze che hanno un tempo_di_preparazione maggiore. In entrambe le collection il campo prezzo identifica il prezzo base del prodotto.

- **Listino_aggiunte**

-nome	string
-prezzo	number
-descrizione	string
-pizza_bibita	boolean

Il campo pizza_bibita identifica se l'aggiunta è relativa alle bibite oppure alle pizze secondo la seguente logica:

- TRUE -> pizza
- FALSE -> bibita

- **Ordini**

-tavolo	number
-persone_da_servire	number
-ordine_completato	boolean
-stato_bibite	number
-stato_pizze	number
-ordine_pagato	boolean
-ora_ordine	Date
-cameriere	string
-barista	string
-cassiere	string

Questo schema raffigura tutto il ciclo di vita di un ordinazione presa da un cameriere, comprende il tavolo dove sono seduti i clienti, quanti clienti bisogna servire(utile a fini statistici), il cameriere che ha preso l'ordinazione e l'orario che è stata effettuata l'ordinazione.

Il campo ordine_completato indica se l'ordinazione è stata conclusa ovvero se tutti i clienti del tavolo hanno ordinato i prodotti da loro scelti, in caso affermativo i prodotti (pizze + bibite) verranno commissionate ai pizzaioli e baristi.

I campi stato_bibite e stato_pizze sono dei campi numerici che memorizzano lo stato attuale delle rispettive bibite o pizze. I valori che esse possono assumere sono:

Valore campo stato_bibite:

0. Bibite non pronte
1. Bibite in lavorazione
2. Bibite pronte
3. Bibite consegnate

Valore campo stato_pizze:

0. Pizze non pronte
1. Pizze pronte
2. Pizze consegnate

Il campo barista contiene l'username dell'addetto che ha preparato tutte le bibite dell'ordinazione, questo campo può essere memorizzato in questo schema perché un solo barista prepara tutte le bibite di un tavolo.

N.B: Più pizzaioli invece possono preparare le pizze di un'unica ordinazione, quindi le informazioni su chi ha preparato la pizza o se la pizza è in lavorazione verranno salvate sui singoli prodotti ordinati.

Il campo ordine_pagato indica se l'ordine è stato pagato dai clienti e di conseguenza libera logicamente il tavolo.

Infine al termine dell'operazione di pagamento viene assegnato al documento il cassiere che ha generato lo scontrino.

• Prodotti_ordinati

-nome_prodotto	string
-aggiunte	[string]
-prezzo	number
-pizzaiolo	string
-bibita_pizza	boolean
-ordine	string
-stato_preparazione	number
-tempo_di_preparazione	number

I documenti della collezione **Prodotti_ordinati** vengono creati ogni qual volta che viene aggiunto un prodotto ordinato inerente ad una ordinazione, ogni prodotto ordinato rimane dunque "collegato" al suo documento **Ordini** attraverso il campo ordine che contiene il numero dell'ordinazione(`_id`).

Lo schema memorizza quindi il nome del prodotto base, un array contenente le aggiunte e il prezzo del prodotto (prezzo base del prodotto + prezzo aggiunte).

Un booleano bibita_pizza identifica se il prodotto è una bibita o pizza secondo la seguente logica:

- TRUE -> bibita
- FALSE -> pizza

Di conseguenza come accennato già in precedenza s.s.e. il documento in questione è una pizza ad esso verranno memorizzati anche il pizzaiolo che sta o ha preparato la pizza, lo stato di preparazione della pizza, e il tempo di preparazione della pizza che è uguale al tempo di preparazione della pizza base contenuto in **Listino_pizze**.

Volendo si avrebbe potuto pensare di utilizzare un valore speciale su un campo inerente solo alle pizze per distinguere se il prodotto era una bibita od una pizza ma avrebbe complicato la logica dello schema e della selezione tra pizze e bibite, quindi di fatto si è optato per aggiungere un campo booleano per la distinzione dei due prodotti.

Valore campo stato_preparazione:

0. Pizza non pronta
1. Pizza in lavorazione
2. Pizza pronta

Lista endpoints

Quasi tutti gli endpoint sono accessibili solo da un particolare ruolo, tranne alcuni che si possono definire endpoint base e non è necessario un ruolo particolare per accedervi.

Endpoint accessibili senza ruolo

Nome	Metodo	Parametri	Descrizione
/	GET	-	Restituisce la versione delle api e un vettore di stringhe contenenti tutti gli endpoint.
/listino_pizze	GET	-	Ritorna il listino pizze
/listino_bibite	GET	-	Ritorna il listino bibite
/listino_aggiunte_pizze	GET	-	Ritorna il listino delle aggiunte per le pizze
/listino_aggiunte_bibite	GET	-	Ritorna il listino delle aggiunte per le bibite
/elenco_tavoli	GET	-	Ritorna l'elenco dei tavoli
/renew	GET	-	Restituisce un JWT appena generato, previo login
/login	GET	-	Effettua il login se l'utente esiste, ritorna un JWT

Endpoint accessibili per ruolo Cameriere

Nome	Metodo	Parametri	Descrizione
/crea_ordine	POST	-	Crea un nuovo ordine.
Body richiesta			
{ tavolo : 5 , persone_da_servire : 10 }			
/aggiungi_prodotto/	POST	id_ordine	Aggiungi un prodotto ad una ordinazione.
Body richiesta			
{ nome_prodotto : "CocaCola" , aggiunte : ["ghiaccio", "limone"] }			
/chiudi_ordine/	POST	id_ordine	Chiudi ordinazione.
/ordini_da_consegnare	GET	-	Restituisce una lista di ordini effettuati dal cameriere che chiama l'api, ai quali le bibite o le pizze od entrambe sono pronte per la consegna.
/bibite_consegnate/	POST	id_ordine	Segnala l'avvenuta consegna di tutte le bibite dell'ordine inviato come parametro
/pizze_consegnate/	POST	Id_ordine	Segnala l'avvenuta consegna di tutte le pizze dell'ordine inviato come parametro

Endpoint accessibili per ruolo Barista

Nome	Metodo	Parametri	Descrizione
/prossima_ord_barista	GET	-	Restituisce il numero del tavolo e la lista delle bibite da preparare dell'ordinazione effettuata da più tempo.
/abbandona_ordine_bibite/	POST	id_ordine	"Sblocca" un'ordinazione non compiuta che era stata prelaionata da un barista. L'ordinazione ritorna ad essere accessibile ad eventuali baristi che effettuano la chiamata all'api /prossima_ord_barista
/bibite_pronte/	POST	id_ordine	Segnala al cameriere che aveva effettuato l'ordinazione che tutte le bibite della stessa sono pronte per la consegna.

Endpoint accessibili per ruolo Pizzaiolo

Nome	Metodo	Parametri	Descrizione
/prossima_ord_pizzaiolo	GET	-	Restituisce il numero del tavolo e la lista delle pizze da preparare dell'ordinazione effettuata da più tempo.
/pizza_inizio_preparazione/	POST	id_prodotto	Segnala a tutti gli altri pizzaioli che la pizza è stata infornata.
/pizza_fine_preparazione/	POST	id_prodotto	Segnala a tutti gli altri pizzaioli che la pizza è stata sfornata.
/pizze_pronte/	POST	id_ordine	Segnala al cameriere che aveva effettuato l'ordinazione che tutte le pizze della stessa sono pronte per la consegna.

Endpoint accessibili per ruolo Cassiere

Nome	Metodo	Parametri	Descrizione
/prossima_ord_pizzaiolo	GET	-	Restituisce il numero del tavolo e la lista delle pizze da preparare dell'ordinazione effettuata da più tempo.
/pizza_inizio_preparazione/	POST	id_prodotto	Segnala a tutti gli altri pizzaioli che la pizza è stata infornata.
/pizza_fine_preparazione/	POST	id_prodotto	Segnala a tutti gli altri pizzaioli che la pizza è stata sfornata.
/pizze_pronte/	POST	id_ordine	Segnala al cameriere che aveva effettuato l'ordinazione che tutte le pizze della stessa sono pronte per la consegna.
/tavoli_occupati	GET	-	Restituisce una lista dei tavoli occupati.
/informazioni_ordine/	GET	numero_tavolo	Restituisce tutto il documento ordini del tavolo inviato come parametro.
/informazioni_prodotti/	GET	id_ordine	Restituisce tutti i documenti prodotti_ordinati inerenti all'ordine inviato come parametro.

Nome	Metodo	Parametri	Descrizione
/calcolo_totale/	GET	numero_tavolo	Calcola e restituisce il totale da pagare del tavolo inviato come parametro.
/pagamento_ordine/	POST	numero_tavolo	Segnala l'avvenuto pagamento del tavolo inviato come parametro. Inoltre imposta il cassiere che ha ultimato il pagamento, ed infine setta il tavolo stesso come libero.
/incasso_giorno	GET	-	Restituisce l'incasso odierno
/incasso_settimana	GET	-	Restituisce l'incasso degli ultimi 7 giorni.
/incasso_mese	GET	-	Restituisce l'incasso degli ultimi 7 giorni.
/users_camerieri	GET	-	Ritorna una lista dei camerieri.
/users_baristi	GET	-	Ritorna una lista dei baristi.
/users_pizzaioli	GET	-	Ritorna una lista dei pizzaioli.
/users_cassieri	GET	-	Ritorna una lista dei cassieri.
/statistiche_dipendente/	GET	id	Restituisce delle statistiche differenti a seconda del ruolo del dipendente passato come parametro.

Autenticazione degli utenti

L'autenticazione degli utenti è gestita interamente utilizzando i metodi: basic authentication e bearer token.

N.B: Tali metodi non garantiscono però una comunicazione sicura e non garantiscono l'autenticità delle parti, per fare ciò è opportuno utilizzare https (protocollo per la comunicazione sicura).

Nel database MongoDB per ovvi motivi le password degli utenti non sono salvate in chiaro ma bensì utilizzando due stringhe **digest** e **salt**.

Quando vi è la registrazione di un nuovo utente viene creata una stringa random di 32 caratteri esadecimali -> **salt**

Dopodiché viene creata un'istanza Hmac utilizzando come funzione hash: "sha256" e come chiave il **salt** creato in precedenza.

All'istanza Hmac verrà poi aggiunta la password che l'utente ha inserito e verrà dunque calcolato il **digest** esadecimale.

Workflow Login:

1. Inizialmente l'utente per autenticarsi dovrà fornire username e password che verranno inviate criptate in base-64 tramite basic authentication.
2. Il server quindi procede con una strategia basic authentication definita mediante il middleware Passport, tale strategia consiste prima di verificare se l'utente esiste e poi in caso affermativo verificare la password immessa.

Per verificare la password ricorreremo all'utilizzo di salt e digest come visto in precedenza.

3. Inizialmente verrà creato l'istanza hmac con il **salt** creato al momento della registrazione.
4. Alla stringa Hmac verrà poi aggiunta la password appena immessa dall'utente e verrà dunque calcolata la stringa esadecimale **digest**.
5. Il nuovo **digest** viene dunque confrontato con quello creato al momento della registrazione dell'utente, se uguali significa che la password immessa dall'utente è corretta.
6. In caso di digest omogenei il server quindi procederà creando il JWT con le informazioni dell'utente (username, cognome, nome, ruolo), provvederà poi a firmarlo e inviarlo come risposta all'utente.

L'utente potrà ora navigare nelle pagine della web app utilizzando come metodo di autenticazione bearer token passando per l'appunto il JWT. Il server quindi attraverso il JWT conosce il ruolo dell'utente e può dunque negare eventuali richieste ad endpoint non accessibili per quel ruolo.

Client web

La web application è stata sviluppata con il framework Angular. Angular è una piattaforma open source per lo sviluppo in TypeScript di applicazioni web, la sua principale particolarità risiede nel fatto di poter scrivere l'applicazione una volta e portarla su diversi tipi di dispositivo mediante dei toolkit.

Nel nostro caso abbiamo utilizzato Electron e Cordova per creare rispettivamente un'applicazione desktop Windows e un'applicazione mobile android.

Questo porting è stato però permesso grazie a Bootstrap, framework grafico che permette la realizzazione di design responsivo, ovvero la grafica del sito si adatta in funzione alle dimensioni del dispositivo utilizzato.

L'intera applicazione Angular può essere divisa in Components: pagine accessibili nel client web, e Service: file contenente metodi.

Components:

- **barista:** Pagina dedicata al barista nella quale è visualizzato l'ordine da preparare con il relativo numero di tavolo e la lista di tutte le bibite.
- **pizzaiolo:** Pagina dedicata al pizzaiolo nella quale è visualizzato l'ordine da preparare con il relativo numero di tavolo e la lista di tutte le pizze.
- **cameriere:** Pagina dove il cameriere crea le ordinazioni selezionando il tavolo appena occupato dai clienti.
- **cameriere-prende-ordinazioni:** Pagina dove il cameriere inserisce le bibite ordinate dal cliente di un tavolo.
- **cameriere-prende-ordinazioni2:** Pagina dove il cameriere inserisce le pizze ordinate dal cliente di un tavolo.
- **cameriere-consegna-ordinazioni:** Attraverso questa pagina il cameriere visualizza le ordinazioni alle quali le bibite o le pizze od entrambe sono pronte per la consegna.
- **cassiere:** Pagina principale adibita al cassiere, in questa pagina è possibile visualizzare, stato di ogni singolo tavolo, stato delle bibite, stato delle pizze e altre varie statistiche inerenti ad un'ordinazione. È inoltre possibile visualizzare il totale e pagare il conto.
- **cassiere-gestione-utenti:** Attraverso questa pagina il cassiere visualizza una lista di tutti i dipendenti registrati sulla pizzeria. È possibile selezionare un'utente e visualizzare le sue relative statistiche, inoltre è possibile eliminare l'utente stesso.
- **cassiere-incasso:** Pagina che permette di visualizzare l'incasso giornaliero, settimanale e mensile della pizzeria. È accessibile solo dai cassieri.
- **user-login:** Pagina principale della web app ove è possibile effettuare il login.
- **user-signup:** Attraverso questa pagina è possibile registrare un nuovo utente, la pagina è accessibile solo dai cassieri.

N.B: I collegamenti tra le varie pagine sono illustrati nel **Workflow dell'applicazione.**

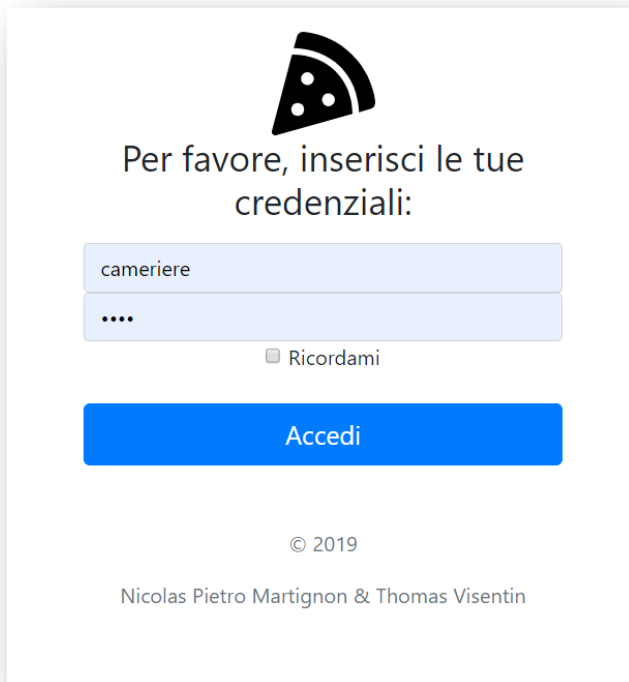
Services:

- **prodotti.service:** File che racchiude tutte le chiamate alle api del server riguardanti la gestione dei prodotti e delle ordinazioni.
- **user.service:** File che racchiude metodi riguardanti la gestione degli utenti. Alcuni metodi operano in locale restituendo un determinato campo del JWT, altri metodi invece fungono da wrap di una chiamata alla api del server che soddisfa tale richiesta.

Routes:

Component	Route
barista	/barista
pizzaiolo	/pizzaiolo
cameriere	/cameriere
cameriere-prende-ordinazioni	/cameriereOrd
cameriere-prende-ordinazioni2	/cameriereOrd2
cameriere-consegna-ordinazioni	/cameriereCons
cassiere	/cassiere
cassiere-gestione-utenti	/gestioneutenti
cassiere-gestione-incasso	/incasso
user-login	/login
user-signup	/signup

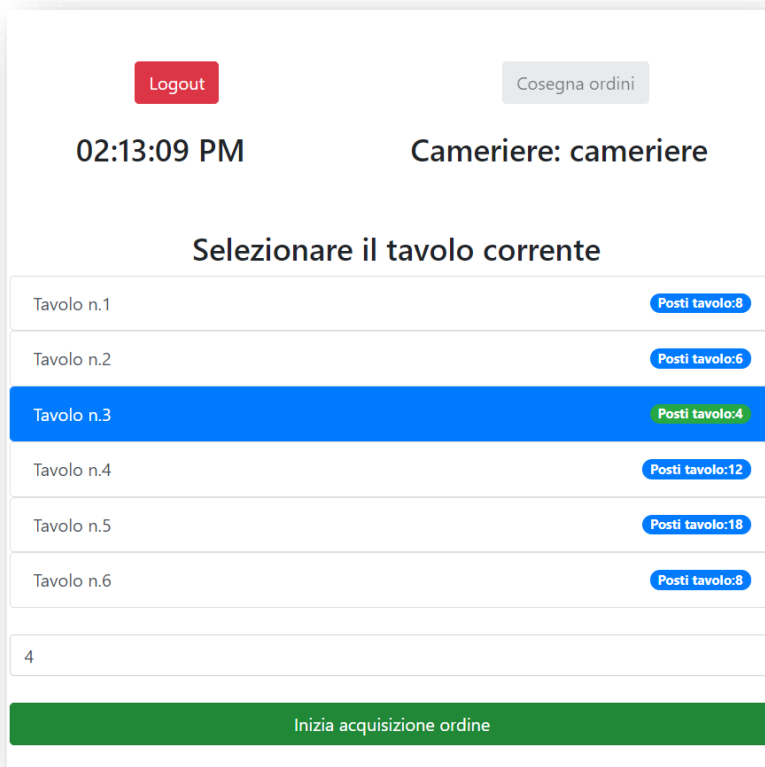
Workflow dell'applicazione



The login screen features a pizza icon at the top. Below it, the text "Per favore, inserisci le tue credenziali:" is displayed. There are two input fields: the first contains the username "cameriere" and the second contains masked characters "....". A checkbox labeled "Ricordami" is positioned below the password field. A large blue button labeled "Accedi" is centered below the inputs. At the bottom, the copyright notice "© 2019 Nicolas Pietro Martignon & Thomas Visentin" is shown.

Component : **user-login**

Il cameriere effettua il login inserendo il suo username, in questo caso "cameriere" scelto da noi per comodità. Dopo aver inserito la password ed aver premuto il pulsante accedi, se le credenziali sono corrette il cameriere verrà reindirizzato nella sua schermata principale (**cameriere**).



The main interface for the waiter includes a top bar with a red "Logout" button and a grey "Cosegna ordini" button. Below this, the current time "02:13:09 PM" and the user name "Cameriere: cameriere" are displayed. The main section is titled "Selezionare il tavolo corrente" and contains a list of tables. Each table entry shows the table number and a button indicating the number of occupied seats. Table n.3 is currently selected, highlighted in blue, with 4 occupied seats. Below the list is a text input field containing the number "4". At the bottom, a large green button labeled "Inizia acquisizione ordine" is present.

Tavolo n.	Posti tavolo:
Tavolo n.1	8
Tavolo n.2	6
Tavolo n.3	4
Tavolo n.4	12
Tavolo n.5	18
Tavolo n.6	8

Component : **cameriere**

In questa pagina il cameriere seleziona il tavolo occupato dai clienti e inserisce il numero di clienti seduti. Premendo il pulsante verde in basso si dà inizio quindi ad una nuova ordinazione procedendo nella pagina (**cameriere-prende-ordinazioni**).

Tavolo N°: 3 Numero persone: 4

Ordinazione bibite

Articoli ordinati:

CocaCola	Quantità: 2
Ghiaccio Limone	
Birra piccola	Quantità: 2

Chiudi ordinazione bibite

Listino Articoli :

Aranciata	2.8€
Birra piccola	3€
CocaCola	2.8€

Listino Aggiunte :

Ghiaccio	0€
Limone	0€

Prodotto aggiunto correttamente!

2

Aggiungi

Component : **cameriere-prende-ordinazioni**

Il cameriere inizia dunque a prendere le ordinazioni delle bibite, selezionando dalle apposite liste l'articolo con l'eventuale aggiunta. Dopo aver inserito tutte le bibite il cameriere può chiudere l'ordinazione delle bibite premendo il pulsante giallo.

Si verrà quindi reindirizzati nella schermata (**cameriere-prende-ordinazioni2**).

Tavolo N°: 3 Numero persone: 4

Ordinazione pizze

Articoli ordinati:

Margherita	Quantità: 2
Mozzarella di bufala	
Capricciosa	Quantità: 2

Chiudi ordinazione

Listino Articoli :

Capricciosa	6€
Diavola	6€
Margherita	4€

Listino Aggiunte :

Mozzarella di bufala	1.5€
----------------------	------

Prodotto aggiunto correttamente!

2

Aggiungi

Component : **cameriere-prende-ordinazioni2**

In questa schermata il cameriere prende le ordinazioni delle pizze con la stessa logica della pagina precedente. Dopo aver inserito tutte le pizze il cameriere può chiudere l'ordinazione premendo il pulsante giallo.

L'ordinazione è dunque completata, ora alla coda di preparazione dei baristi e pizzaioli verrà aggiunta l'ordinazione appena effettuata.

Il cameriere verrà reindirizzato nella sua schermata principale (**cameriere**).

Dopo che entrambi barista e pizzaiolo abbiano effettuato il login essi verranno reindirizzati nella loro pagina dedicata.

The screenshot shows the Barista interface. At the top, there is a red 'Logout' button, the time '03:28:16 PM', and the role 'Barista: barista'. Below this, it says 'Ora ordine: 03:20:36 PM' and 'Tavolo: 3'. The main area contains a list of drinks: two 'CocaCola' items, each with 'Ghiaccio' and 'Limone' buttons, and two 'Birra piccola' items. At the bottom, there is a yellow 'Bibite Pronte' button.

Component : **barista**

Nella sua schermata il barista riceve l'ordinazione appena effettuata se la sua coda di produzione era vuota. Dopo aver preparato tutte le bibite il barista preme il pulsante giallo che segnala al cameriere che aveva effettuato l'ordinazione che le bibite del tavolo 3 sono pronte.

The screenshot shows the Pizzaiolo interface. At the top, there is a red 'Logout' button, the time '03:29:24 PM', and the role 'Cameriere: pizzaiolo'. Below this, it says 'Ora ordine: 03:20:36 PM' and 'Tavolo: 3'. The main area contains a list of pizzas: two 'Capricciosa' items, each with a 'T.P. 6' label and an 'Inizio preparazione' button, and two 'Margherita' items, each with a 'Mozzarella di bufala' label, a 'T.P. 4' label, and an 'Inizio preparazione' button.

Component : **pizzaiolo**

Nella sua schermata il pizzaiolo riceve l'ordinazione appena effettuata se la sua coda di produzione era vuota. Dopo che ha preparato tutte le pizze il sistema segnalerà al cameriere che aveva effettuato l'ordinazione che tutte le pizze del tavolo 3 sono pronte.

Entrambe le schermate dopo aver preparato i prodotti di un'ordinazione verranno riempite con la ordinazione successiva da preparare se la coda di produzione non è vuota.

Dopo che i prodotti di un tavolo sono stati preparati il pulsante “Consegna ordini” nel component **cameriere**, che inizialmente era grigio diventerà giallo ed inizierà a pulsare segnalando al cameriere che vi sono delle ordinazioni da consegnare.



Se premuto il cameriere verrà reindirizzato nella schermata (**cameriere-consegna-ordinazioni**).

Component : **cameriere-consegna-ordinazioni**

In questa pagina il cameriere può dunque segnalare che tutte le pizze o tutte le bibite di un'ordinazione sono state consegnate sul loro tavolo di appartenenza.

Logout
08:29:54 PM
Cassiere: capo

Incasso
Gestione utenti

Gestione tavoli:

1
2
3
4
5
6

Tavolo: 3
Cameriere assegnato: cameriere
Barista: barista
Numero persone: 4
Ora ordine: 03:20:36 PM
Stato pizze: **Pizze consegnate**
Stato bibite: **Bibite consegnate**

Pizze ordinate:

Margherita	5.5€	Stato pizza: Preparata	Pizzaiolo: pizzaiolo
Mozzarella di bufala			
Margherita	5.5€	Stato pizza: Preparata	Pizzaiolo: pizzaiolo
Mozzarella di bufala			
Capricciosa	6€	Stato pizza: Preparata	Pizzaiolo: pizzaiolo
Capricciosa	6€	Stato pizza: Preparata	Pizzaiolo: pizzaiolo

Bibite ordinate:

CocaCola	2.8€
Ghiaccio Limone	
CocaCola	2.8€
Ghiaccio Limone	
Birra piccola	3€
Birra piccola	3€

Totale: 34.6€
Crea scontrino e paga

Component : **cassiere**

Il ciclo di vita di un'ordinazione si conclude nella pagina dedicata al cassiere. In questa schermata l'addetto può visualizzare statistiche e creare lo scontrino per ogni tavolo, visualizzare gli incassi (**cassiere-gestione-incasso**) e gestire i dipendenti della pizzeria(**cassiere-gestione-utenti**).

<- Indietro
02:25:35 PM
Cassiere: capo

Statistiche incasso

Incasso giorno : 34.6
Incasso settimana : 34.6
Incasso mese : 34.6

Component : **cassiere-gestione-incasso**

Attraverso questa pagina il cassiere può visualizzare gli incassi giornalieri, degli ultimi 7 giorni oppure degli ultimi 30 giorni.

< - Indietro

Crea nuovo utente

08:31:10 PM

Cassiere: capo

Camerieri:

Username: cameriere

Nome Cognome: Mario Verdi

Elimina

Pizzaioli:

Username: pizzaiolo

Nome Cognome: Mario Visentin

Elimina

Baristi:

Username: barista

Nome Cognome: Mario martignon

Elimina

Cassieri:

Username: capo

Nome Cognome: Mario Rossi


Elimina

Statistiche

- Persone a cui e stato preparato da bere : 4
- Singoli prodotti preparati : 4
- Tavoli a cui e stato preparato da bere : 1

Component : **cassiere-gestione-utenti**

Attraverso questa schermata il cassiere ha un quadro generale di tutti i dipendenti, suddivisi in base al loro ruolo. Per ogni dipendente sono disponibili delle statistiche che differiscono da ruolo a ruolo. È inoltre possibile eliminare un determinato utente, oppure crearne uno nuovo mediante il tasto verde in alto a destra, che reindirizzerà nel component (**user-signup**).



Registrazione nuovo utente

Username

NikoMac

Cognome

Martignon

I tuoi dati personali non verranno condivisi con nessuno.

Nome

Nicolas Pietro

I tuoi dati personali non verranno condivisi con nessuno.

Ruolo

Cassiere

Password

....

Registrati!

Component : **user-signup**

Schermata adibita alla registrazioni di nuovi utenti/dipendenti accessibile solo ai cassieri.