Tecnologie utilizzate

- **HTML** per la strutturazione semantica delle View;
- **CSS** per l'implementazione dell'interfaccia grafica delle View;
- **JavaScript** per la creazione della logica dell'applicazione web;
- **Node.js** per l'esecuzione di JavaScript lato server;
- **Express.js** come framework per la creazione delle API;
- **EJS** come template engine;
- **Passport.js** come middelware per l'autenticazione degli attori;
- **MongoDB** come database per il salvataggio dei dati;
- **API Event Source** per la gestione delle notifiche**.**

Casi d'uso implementati

Login

Registrazione cliente e cuoco

Cliente

1) Creazione di un ordine dal menù;
2) visualizzazione stato dell'ordine.

Cuoco

1) Presa in carico di un ordine;
2) conclusione di un ordine;
3) visualizzazione degli ordini attivi.

Amministratore

1) Aggiunta elemento al menù;
2) eliminazione elemento dal menù;
3) modifica elemento del menù;
4) visualizzazione delle transazioni.

Logout

Note tecniche

Login

L'utente Amministratore è univoco all'interno del sistema e le sue credenziali risultano salvate preventivamente all'interno del database.

Per la gestione criptata della password è stata utilizzata la funzione di hashing *bcrypt* all'interno di un pre hook, rispetto al metodo save() che effettua il salvataggio dell'oggetto, che intercetta la password prima del salvataggio nel database e calcola un hash utilizzando un salt di 10 bit.

Il login è stato gestito attraverso il middleware *Passport.js* e in particolare è stata utilizzata una tipologia di login con strategia locale *local-signup*; come campo username è stato impostato email, inserita dall'utente in fase di registrazione; all'inserimento delle credenziali per effettuare il login verrà effettuato il confronto tra le credenziali inserite e quelle presenti nel database mediante *bcrypt.compare()*, in caso di mancata di corrispondenza si viene automaticamente rimandati alla homepage.

Registrazione cliente e cuoco

La registrazione del cliente avviene tramite la view *registrazione.ejs*, la quale è accessibile dalla homepage; la registrazione del cuoco è effettuabile solo dall'Amministratore e quindi la view *registrazioneChef.ejs* è accessibile tramite l'apposito bottone in *admin.ejs*.

Una volta inseriti i dati e confermato il loro invio viene effettuata una richiesta POST a */users/* che richiamerà il controller *addUser* effettuando l'inserimento dell'utente all'interno del database.

Cliente

La creazione dell'ordine da parte di un Cliente avviene all'interno della view *ordine.ejs*, renderizzata dal controller *getAllProducts* in seguita ad una richiesta GET partita a seguito del corretto login dell'utente come Cliente; man mano che i prodotti vengono selezionati, questi vengono memorizzati all'interno di un vettore *products* tenendo traccia del totale con *sum*; questa logica è gestita nello script *orderScript.js* memorizzato nella cartella *public* e, una volta completato l'ordine, il cliente clicca sull'immagine posizionata accanto al totale dell'ordine: questa operazione fa partire il *submit* collegato all'azione */orders/pagamento*, con cui viene richiamato il controller *pay* il quale gestisce il pagamento simulato richiamando la view *pagamento.ejs*; una volta inseriti i dati di pagamento e dato conferma viene effettuata una richiesta POST al *http://localhost:3000/orders* il cui *body* conterrà l'oggetto ordine costruito in precedenza e che conterrà anche il riferimento all'utente e la data di creazione e che renderizzerà la view *riepilogoOrdine.ejs*.

Nella view *riepilogoOrdine.ejs*, all'utente verranno mostrati i dati relativi all'ordine appena effettuato e, in fondo alla pagina, un modo di tenere traccia dello stato di avanzamento dell'ordine; l'avanzamento dell'ordine viene gestito utilizzando dei SSE che partono dall'endpoint */updates/tocustomer* in cui vengono inviati eventi che indicano lo stato attuale dell'ordine; la view resta in ascolto di eventi mediante script *orderUpdate.js* utilizzando EventSource e quando viene aggiornato lo stato dell'ordine vengono modificati i box in fondo alla pagina dal punto di vista grafico.

Cuoco

L'utente cuoco deve poter gestire gli ordini dalla view *vediOrdini.ejs* quindi prenderli in carico e completarli. Gli ordini sono memorizzati di default con stato *Da prendere in carico* nel database quindi le interazioni del cuoco modificheranno lo stato dell'ordine, ad ogni interazione verrà inviata una notifica SSE verso il cliente che vedrà lo stato del suo ordine aggiornato. Ogni interazione rimanda al controller PATCH *updateOrder* di modifica dell'ordine che si occuperà di aggiornare lo stato e di modificare l'idCuoco ovvero l'ordine conserverà anche il riferimento al cuoco che lo ha gestito.

Quando il cliente effettua un ordine (precisamente quando completa il pagamento simulato) verrà inviata una notifica al cuoco che potrà accettare l'aggiornamento degli ordini da prendere in carico oppure potrà rifiutare la notifica e lavorare sugli ordini attualmente disponibili e in lavorazione. Il cuoco resta in ascolto mediante EventSource verso l'endpoint */updates/tocook* questa logica è gestita nello script *newOrder.js.*

Amministratore

A fronte delle credenziali di Amministratore riconosciute nel login presente in *homepage.ejs*, si viene reindirizzati, con una GET fatta a *http://localhost:3000/products*, al controller *getAllProducts* il quale effettua una query sul database ottenendo tutti i documenti (i prodotti) e renderizza la view *admin.ejs* passando anche il vettore *orders* contenente il risultato della query.

In *admin.ejs* l'utente autenticato come Amministratore può effettuare l'aggiunta di un nuovo prodotto attraverso l'apposito form inserendo nome, prezzo e tipo del prodotto: confermando l'aggiunta si effettua una richiesta POST a */products* richiamando il controller *addProduct* che, dopo aver provveduto all'aggiunta del prodotto all'interno dell'apposita collezione del database ed effettua il redirect verso la route */products*, in breve effettua il refresh della pagina rendendo visibile l'inserimento del prodotto.

La modifica del prodotto viene gestita nella view *admin.ejs*. Può essere selezionato un prodotto da modificare scrivendolo o selezionandolo dal datalist, questa operazione è necessaria per conservare un riferimento al prodotto da modificare. In seguito è possibile inserire il nuovo nome e il nuovo prezzo del prodotto, non è obbligatorio inserirli entrambi. Al click del *conferma* verrà effettuata una POST all'endpoint */products/<%= element._id %>?_method=PATCH* che verrà sottoposta ad override con il metodo PATCH che quindi attiverà la route che riporta al controller *updateProd.* Terminata la modifica verrà effettuato il refresh della pagina con la stessa modalità articolata per l'inserimento.

La cancellazione del prodotto è gestita nella view *admin.ejs*. Può essere selezionato un prodotto da cancellare dal menù semplicemente cliccando l'icona di eliminazione presente per ogni elemento presentato. Il click sull'icona riporta a */products/<%= element._id %>?_method=DELETE* che, nella stessa modalità articolata per la modifica del prodotto, attiverà il controller *deleteProd* al termine del quale verrà effettuato il refresh della pagina con la stessa modalità articolata per l'inserimento.

Attraverso l’apposito tasto è possibile accedere allo storico delle transazione, in particolare viene effettuata una richiesta GET a *orders/all/:opType*; in opType viene inserito il parametro per fare eseguire la reindirizzazione corretta al controller *getAllOrders* che, una volta effettuata la query per ottenere tutti gli ordini effettuati, renderizza *transazioni.ejs* passando l’oggetto *orders* utilizzato nella view per mostrare *idUtente*, riferimento temporale e totale pagato; è possibile tornare alla view *admin.ejs* attraverso una *X* in alto a destra che richiama *history.back()* al click con il mouse.

Logout

L’admin nella sua schermata principale, il client nella schermata di riepilogo ordine e il cuoco nella sua schermata principale possono effettuare il logout in ogni momento cliccando sulla *X* presente nelle interfacce indicate, il click rimanderà alla route */users/logout* che gestirà il logout effettuato mediante Passport in cui verrà resettata la sessione.