Lorenzo Dentis, st914833
Relazione del progetto finale, anno 2020/21
Servizio Gambrinus

Tema del sito
Il sito che ho creato è un po' "particolare" in quanto più che essere un sito web pensato per essere inserito in un contesto di web-browsing è un applicativo adatto a svolgere una specifica funzione all'interno di una rete locale.
In questo paragrafo fornirò un po' di dettagli che ,per quanto non siano implementativi, chiariscono molte scelte di implementazione e di design che ho compiuto.
Questo sito nasce per una mia necessità reale, quella di creare un applicativo per la gestione degli ordini di una birreria, il Gambrinus di Avigliana; è da loro che ho infatti ottenuto il menù ed i loghi (previa autorizzazione della propietaria).
Sono sicuro che in commercio esistono molti programmi atti a soddisfare questa necessità, però questo progetto volevo svilupparlo da più di un anno ed il corso di Tweb mi ha fornito le conoscenze e l'opportunità di prototipare questa mia idea; magari questa idea susciterà interesse e continuerò a svilupparla.
Il sito è quindi molto sintetico e spoglio, per due ragioni: in primo luogo è pensato per essere usato in un ambiente di lavoro intenso, dove è necessario avere le informazioni importanti in evidenza, senza doverle andrare a cercare e riducendo al minimo la confusione creata da animazioni e fronzoli grafici.
In secondo luogo mi è sembrato corretto rimanere coerente con il design dell'azienda, un design molto semplice con loghi in bianco e nero, a volte accompagnati da giallo e verde.
Naturalmente ho comunque adattato questo progetto alle richieste dell'esame, ad esempio se fosse stato solo un progetto personale non avrei inserito il login e avrei optato per una visualizzazione mobile piuttosto che desktop.
Sono presenti solo 3 sezioni, una riservata al login, una a chi lavora in cucina ed una a chi lavora come cameriere.
La più semplice a livello di funzionalità è quella riservata alla cucina, vengono mostrate le comande in ordine di arrivo (nella realtà questi ordini sarebbero stampati) e l'unica operazione possibile è l'eliminazione di un ordine quando questo è concluso.
La parte più complessa del sito è quella dedicata alla sala, lì è possibile sia visualizzare gli ordini che devono essere serviti che prendere nuove comande tramite una navbar, in tale pagina gli ordini possono essere creati, modificati o eliminati (ad eccezione di quelli già inviati in cucina).
La navbar è composta da una serie di dropdown menù, uno per ogni categoria di piatto/ bevanda ordinabile.

Login
Il login è la index page, nonchè la pagina a cui si viene reindirizzati in caso si cerchi di accedere senza essersi autenticati o dopo il logout.
Il codice JS associato a questa pagina comunica esclusivamente con una pagina, login.php che implementa le funzioni di registrazione e di autenticazione
Tramite un pulsante in alto a destra è possibile richiedere la registrazione di un nuovo utente, a quel punto apparirà una Box in cui è possibile inserire i propri dati.
Se l'utente lascia campi vuoti la richiesta AJAX non viene inviata, qualsiasi altro controllo sui dati viene svolto nel backend.
Se si verifica un errore di qualsiasi tipo viene mostrato un paragrafo di errore che specifica qual'è il problema in modo semplice e generico eventualmente dicendo come può essere risolto.

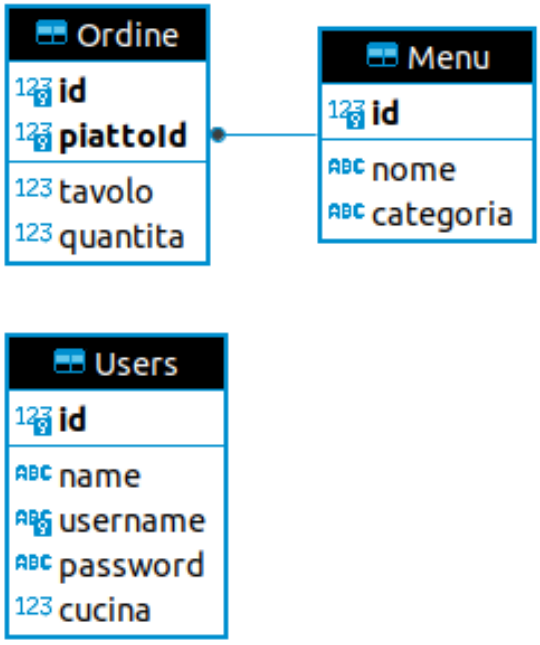
Una volta che l'utente è connesso (quindi viene avviata una nuova sessione) viene automaticamente reindirizzato verso la pagina corretta a seconda del suo ruolo, cucina o sala. In ogni caso viene anche controllato dal server che l'utente non stia cercando di bypassarre il controllo (ad esempio un utente Cucina che cerca di accedere alla pagina Sala o viceversa).
Entrambe le richieste vengono attraverso una POST request via AJAX, uno dei parametro è infatti "login" il cui campo può essere "new" se l'utente si sta registrando oppure "existing" se l'utente si sta loggando.
A seconda di quello che l'utente ha richesto invierà diversi parametri nella Request: nel caso si un utente già registrato solo username e password, se invece si tratta di un nuovo utente anche un nome con cui identificarlo ed il suo ruolo (identificato dalle stringhe "cook" o "waiter")
La prima cosa fatta nel file login.php è un controllo dei parametri POST, che se errati restituiscono una Response di errore (guardare la sezione Struttura delle Response di questo documento)
Successivamente la eventuale sessione precedente viene distrutta e ne viene creata una nuova. A seconda del parametro login vengono quindi chiamate due funzioni del file database.php che si occupa di tutte le operazioni riguardanti il db.
Nel caso della registrazione i campi name e username vengono passati attraverso la funzione strip_tags per prevenire XSS e in entrambe le funzioni viene effettuato lo sha256 della password e tutti i campi vengono passati attraverso la funzione quote per prevenire sql injection.
A questo punto l'utente viene aggiunto al db o se ne verificano le credenziali; in caso di errore viene restituita una response di errore dipendete dal tipo di problema (password errata, utente già esistente, etc...), in caso di successo viene restituito un JSON al cui interno è presente il "ruolo" dell'utente, che serve allo script JS per stabilire a che pagina reindirizzare il browser.

Cucina
Questa pagina viene caricata in seguito al login di un membro della cucina,viene immediatamente eseguito un controllo sulla sessione per assicurarsi che l'utente sia autenticato e abbia il ruolo "cook", altrimenti l'utente viene reindirizzato al login.
A questo punto Javascript si occupa di chiedere al server la lista delle comande, operazione che viene ripetuta ogni 10 secondi per controllare l'arrivo di nuovi ordini.
Tutte le richieste provenienti dagli script kitchen.js e serving.js sono indirizzate ad una pagina: staffRequest.php che funge da wrapper per le richieste controllando la correttezza di queste e identificando l'azione da compiere.
I parametri da inviare nella post request dipendono molto dal tipo di azione ma in linea di massima bisogna sempre inviare un campo "action" (che può essere delete o orders etc...) seguito dall' eventuale oggetto su cui tale azione deve essere effettuata. Ad esempio per eliminare l'ordine #66 bisognerà includere come parametri action=delete id=66.
Se la richiesta non è corretta viene inviato un JSON di errore strutturato secondo le regole descritte nel paragrafo Struttura della Response, se invece la richiesta è formattata correttamente vengono richiamate le corrette funzioni nel file database.php.
Nel caso in cui la richiesta sia di eliminazione non viene ritornato alcun risultato particolare, il client in automatico quando verifica l'assenza di errori elimina l'ordine.
In caso contrario (il client ha chiesto un refresh degli ordini) gli viene restituita una lista di tutte le comande non ancora concluse, sarà poi il client a controllare se ci sono nuovi ordini da visualizzare.
In pratica viene effettuata una semplice query: "select * from orders", il cui risultato viene trascritto come array associativo in una response json.
Gli ordini sono mostrati in paragrafi separati da una dottet line, in ogni paragrafo è mostrato il tavolo ed una tabella recante tutte le informazioni necessarie a completare la comanda; in basso a destra è presente un pulsante per concludere l'ordine.
Il numero di tavolo e il pulsante sono legati ai bordi sinistro e destro della finestra, la dimensione della tabella è invece relativa alla dimensione della finestra in modo da rendere la pagina completamente responsive ed anche utilizzabile da un dispositivo con layout "landscape".

Sala
Questa pagina è la più complessa sia come funzionalità che come aspetto.
È presente una Navbar che viene creata richiedendo al server tutte le categorie di piatti (pizze, panini, etc...).
A parte la richiesta AJAX per ottenere le varie tipologie di piatto è tutto creato tramite CSS, non vi è alcuna funzione JavaScript.
Invece quando l'utente sposta il cursore su una categoria della navbar viene richiesto al server di fornire tutti i piatti appartenenti alla categoria selezionata, a meno che questi non siano già stati richiesti in precedenza.
Tutte le richieste vengono mandate al server nella pagina staffRequest.php (il cui funzionamento è stato descritto in precedenza). Il server si occupa quindi di restituire la lista delle tipologie o la lista dei piatti, dopo aver opportunamente usato la funzione quote per impedire una sql injection tramite modifica dei parametri della post request.
Vi sono poi due colonne, una riservata ai tavoli, l'altra agli ordini.
All'apertura della pagina ed ogni 10 secondi viene effettuato una richiesta al server per stabilire quali tavoli sono "occupati" e quali tavoli sono invece ancora in attesa di ordinare/di essere serviti (anche questo potrebbe essere migliorato, in quanto servirebbe anche avere una lista dei tavoli che hanno ricevuto l'ordine e devono solo più pagare, ma per semplicità ho completamente eliminato tutta la parte dei pagamenti).
Il server risponderà con una lista dei tavoli occupati ed il client si occuperà di spostare di conseguenza i tavoli da occupato a libero o viceversa.
Quando un tavolo viene selezionato dalla colonna di sinistra viene richiesta al server la comanda relativa a quel tavolo, a seconda che il tavolo ne abbia una o meno (occupato o libero) accadono due cose all'apparenza identiche ma molto diverse.
Nel caso il tavolo sia occupato viene semplicemente mostrato il suo ordine nella colonna di destra.
Il tutto viene svolto tramite una richiesta AJAX e la response contiene semplicemente un array in cui ogni elemento è una riga della tabella ottenuta dall'operazione di join fra la tabella "ordine" e "menù" (struttura del DB). In questo modo il client può ottenere i piatti richiesti e la loro quantità.
Nel secondo caso, tavolo libero, viene preparata la colonna di destra per l'inserimento di un nuovo ordine.
Vengono mostrati due nuovi pulsanti per l'invio/cancellazione dell'ordine, quando premuti uno invia la lista di piatti e relative quantità al server, l'altro pulisce la colonna di destra ed elimina la lista.
La composizione del nuovo ordine è tutta gestita dal client.
Quando un tavolo è selezionato (variabile isNewOrder) l'utente può cliccare sui piatti che vuole siano ordinati, qui la funzione addToOrder li inserisce nel dizionario "order" contenente l'id dei piatti come key ed il numero di piatti di tale tipo come value. Ad esempio "42","2" per rappresentare 2 pizze margherita.
Quando il pulsante di invio viene premuto la variabile order viene inserita nel corpo di una POST request ed al server viene chiesto di inserire l'ordine nel database, quando questa operazione viene completata (AJAX success) il browser mostra un messaggio di successo ed i tavoli vengono riaggiornati per mostrare il nuovo tavolo come occupato.
Per poter caricare l'ordine nel database il server prima trasforma il JSON in un array associativo dato che AJAX automaticamente riempie le celle del vettore con valori nulli (se invio due piatti, uno con id 2 e l'altro con id 4 avrò un vettore di 5 elementi; gli elementi [2] e [4] corretti, gli altri 3 elementi nulli) poi passa questo vettore alla funzione newOrder della pagina database.php
Tale funzione calcolo il nuovo id della comanda ed esegue un ciclo su ogni elemento del vettore costruendo una query insert che abbia come values tutti i piatti e le quantità opportunamente formattati (funzione quote).
Se ci sono problemi, quali eccezioni, viene restituito al client un messaggio di errore che si occuperà di mostrarlo all'utente. In ogni caso l'unica situazione in cui tale errore può verificarsi è se l'utente sta cercando di accedere al database in maniera "illecita", quindi il messaggio di errore è

utile alla risoluzione ma anche abbastanza vago per non fornire troppe informazioni ad un eventuale attaccante.
In fondo alla pagina è presente un paragrafo che serve come backfeed all'utente, mostra eventuali messaggi di errore e un messaggio quando un ordine viene spedito correttamente. In generale c'è un solo caso lecito in cui l'utente può effettuare una azione che causa errore (cercare di inserire piatti in un ordine prima di aver scelto il tavolo); se all'utente si presenta qualche altro messaggio significa che c'è un errore di programazzione o l'utente sta falsificando le POST request.
La pagina risulta responsive fino ad una larghezza minima di 720px (quella che in android viene definita risoluzione mdpi); ho imposto questo vincolo per evitare che le tabelle degli ordini diventassero illeggibili ed il contenuto della navbar finisse fuori schermo.
Avrei potuto implementare un design alternativo in modo che il sito risultasse responsive anche per dispositivi molto piccoli, ma mi è sembrato veramente eccessivo e dispendioso.

Struttura del DB
Il database, il cui script di creazione è completamente inserito all'interno del file database.sql, è composto di 3 tabelle:



1. Users: la tabella riservata al login. Ogni utente possiede un id, un nome, un username univoco e l'hash della sua password. In ultimo è presente un bit che identifica se l'utente ha un ruolo in cucina o no, in tal caso è un cameriere.
2. Menu: questa tabella contiene tutti i piatti, allo stesso modo del menù cartaceo. Ogni piatto ha un Id univoco, un nome ed una categoria a cui appartiene.
3. Ordine: Ogni riga della tabella rappresenta un piatto di un ordine; quinidi possiede un Id non univoco, il tavolo a cui è assegnato, l'id del piatto (chiave esterna legata alla tabella menù) e il numero di piatti di quel tipo. Tutte le righe aventi uno stesso Id rappresentano un ordine

Struttura delle Response
Ho cercato di seguire il più possibile il concetto di API e le regole comunemente usate, quindi i codici di errore corretti e dei messaggi più strutturati possibile, a costo di essere "vuoti" o ridondanti.
Ogni response segue questa struttura json

```
{
        info: {..... richiesta inviata dal client....},
        response/error: {....risposta del server o errore}
}
```

nel caso si tratti di una risposta positiva il contenuto di respose varia da caso a caso, nel caso si tratti di un errore la struttura di error è la seguente:

error: {
    title: breve messaggio che indica l’errore
    status: codice di errore coerente con lo standard del documento RFC 2616
    description: descrizione più accurata della causa dell’errore
}
Nelle response di errore non vengono mai inviati codici di errore o messaggi generati dal database (PDOException -> getMessage()), questi vengono solo stampati su eventuale console in quanto potrebbero fornire rilevanti informazioni sul sitema.
Vengono invece inviati messaggi che potrebbero essere utili all’utente o ad un eventuale programmatore che si appoggi a queste API per risolvere l’errore.