

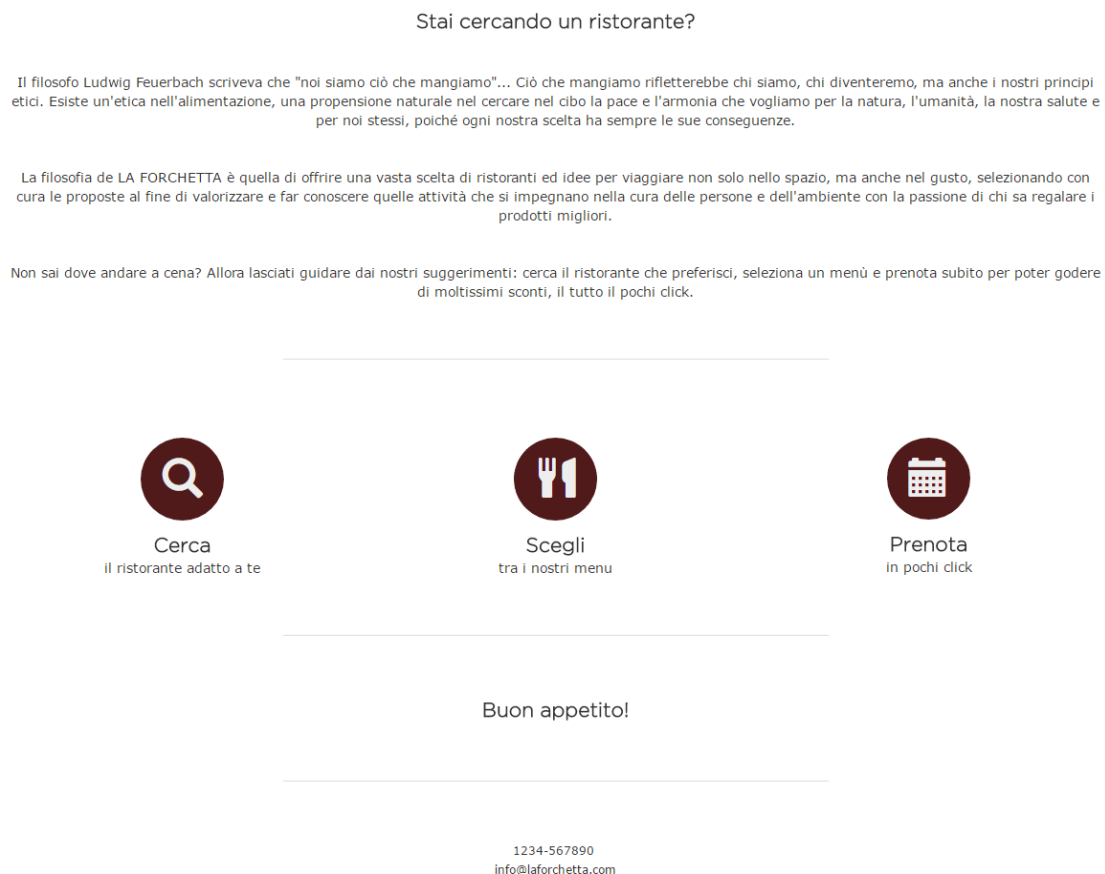
Teresa Tanzi - 858985

APPLICAZIONE WEB

L'applicazione web è composta dalle seguenti schermate:

HOMEPAGE

Pagina iniziale del sito che descrive brevemente il servizio offerto, mostra dei link alle pagine principali del sito ed i contatti.



Funzionalità:

- Se nel Local Storage non è presente l'array JSON contenente le informazioni relative ai ristoranti lo carica

```
function salvaRistoranti() {
    var listaRistoranti=window.localStorage.getItem('ristoranti');

    if (listaRistoranti) {
        //non faccio niente: significa che le informazioni dei ristoranti sono già salvate
    } else {
        //devo salvare la lista dei ristoranti
        caricaRistoranti();
        //var datiRistoranti='[{"codice": "antica-trattoria-del-gallo", "nome": "Antica Tratt
    }
}

function caricaRistoranti() {
    var datiRistoranti='[{ \
        "codice": "antica-trattoria-del-gallo", \
        "nome": "Antica Trattoria del Gallo", \
        "immagine": "images/trattoria-gallo.jpeg", \
```

[...]

```
        "prezzo": 14.00, \
        "sconti": [{ \
            "tipo": "orario", \
            "valore": ["19-21"], \
            "sconto": 1.00 \
        }] \
    }] \
}';

window.localStorage.setItem('ristoranti', escapeHtml(datiRistoranti));
}

function escapeHtml(text) {
    return text.replace(/à/g, '&agrave')
        .replace(/è/g, '&egrave')
        .replace(/ì/g, '&igrave')
        .replace(/ò/g, '&ograve')
        .replace(/ù/g, '&ugrave')
        .replace(/é/g, '&eacute')
        .replace(/'/g, '&#39');
}
```

La funzione “salvaRistoranti” prende dal Local Storage la stringa corrispondente alla chiave “ristorante”. Se questa non esiste (non sono ancora stati salvati i dati dei ristoranti necessari al corretto funzionamento dell’applicazione web), chiama la funzione “caricaRistoranti”.

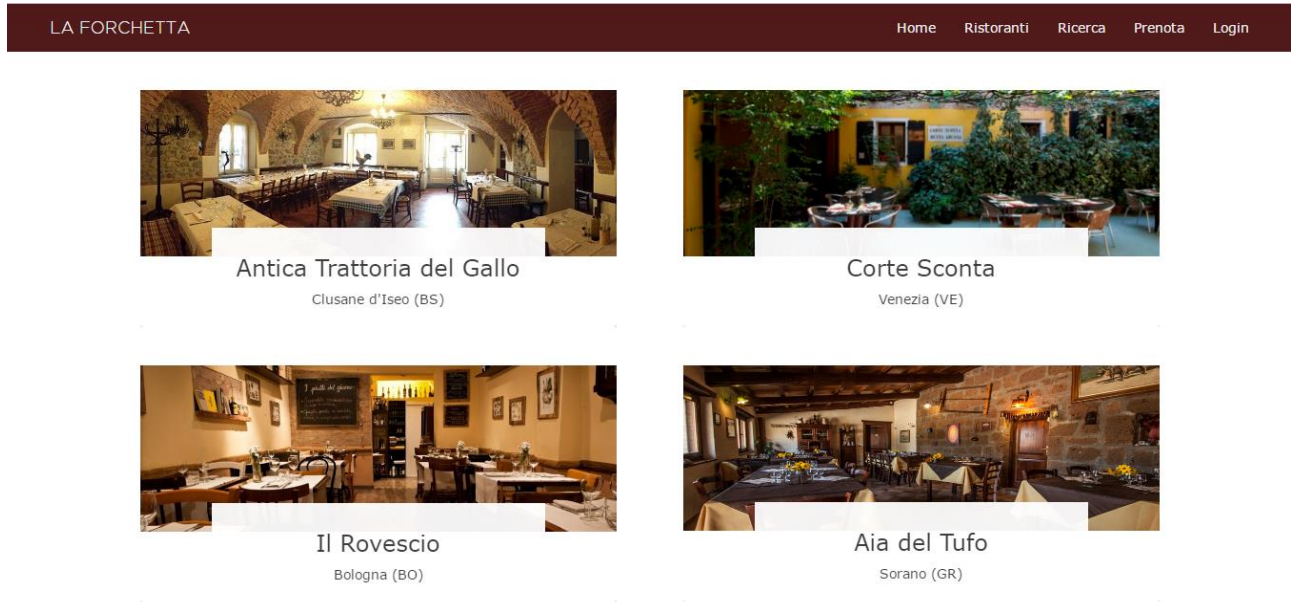
Questa dovrebbe andare a prendere il file JSON contenente le informazioni e caricarlo nel Local Storage, tuttavia, essendo l’applicazione sviluppata in locale, il browser principale utilizzato per lo sviluppo (Google Chrome) impedisce le richieste cross origin e non è stato possibile caricare il file locale tramite una chiamata AJAX. Pertanto la stringa da caricare viene inserita hard coded all’interno del codice.

Prima di venire salvata nel browser si sostituiscono i caratteri accentati e gli apostrofi con i caratteri di escape di HTML, questo per garantire la corretta visualizzazione.

(Le funzionalità della homepage vengono richiamate da tutte le pagine, al fine di avere una corretta rappresentazione dei dati anche se si giunge al sito tramite una pagina differente dalla home)

LISTA RISTORANTI

Lista dei ristoranti contenuti all'interno dell'array JSON. Per ogni ristorante vengono mostrati una foto, il nome, la città e la provincia. Al click su un ristorante si passa alla schermata di dettaglio.



Funzionalità:

- Viene scorso l'array dei ristoranti e per ciascuno viene aggiunto il proprio elemento nella lista
- Ogni ristorante viene rappresentato in una cella della griglia Bootstrap formata da 2 colonne ed un numero non definito di righe (dipende dal numero di ristoranti)

```
var listaRistornti=JSON.parse(window.localStorage.getItem("ristoranti"));

listaRistornti.forEach(function (entry, index) {
  //se l'indice è 0, 2... pari allora apro la riga
  if (index%2==0) {
    $("#lista-ristoranti").append('<div class="row">');
  }

  $("#lista-ristoranti").append('<div class="col-lg-6 col-md-6"> \
    <form action="dettagli_ristorante.html" method="GET" class="image"> \
      <input type="hidden" name="nome" value="'+entry["codice"]+'"> \
      <button type="submit" class="image-submit"> \
        <div class="thumbnail ristorante"> \
           \
          <div class="wrapper"> \
            <div class="caption post-content"> \
              <h3>'+entry["nome"]+'</h3> \
              <p>'+entry["citta"]+" (" +entry["provincia"]+')</p> \
            </div> \
          </div> \
        </div> \
      </button> \
    </form> \
  </div>');

  //se l'indice è 1, 3... dispari allora chiudo la riga
  if (index%2==1) {
    $("#lista-ristoranti").append('</div>');
  }
})
```

Viene recuperata la stringa dei Ristoranti dal Local Storage e viene salvata come array JSON nella variabile listaRistoranti. Questa viene scorsa e, per ogni suo elemento (quindi per ogni ristorante), si aggiunge al div con id “lista-ristoranti” – inizialmente vuoto – un form che passa, in hidden, il codice del ristorante alla pagina dettagli_ristorant.html con metodo GET, quindi tale codice verrà visualizzato all’interno dell’URL.

Il pulsante per inviare il form consiste in un’immagine (una foto del ristorante la cui URL è stata salvata nel Local Storage) ed in un’etichetta contenente alcune informazioni inerenti a quel ristorante.

Rappresentando i ristoranti in una griglia con due colonne è necessario controllare se l’indice all’interno dell’array sia pari o dispari per poter inserire nel codice HTML i tag di inizio e di fine riga nella posizione corretta.

```
.post-content {
  background: #fff;
  opacity: 0.95;
  margin: 0 auto;
  margin-top: -30px;
  text-align:center
  position: relative;
  width: 70%;
  color: #000;
  transition-duration: 1s;
}

.thumbnail {
  margin:0 auto;
  padding: 0;
  border: 0;
}

.thumbnail:hover .post-content {
  width: 100%;
  opacity: 0.95;
  transition-duration: 1s;
}
```

L’etichetta (con classe “post-content”) è leggermente sovrapposta all’immagine grazie alla proprietà position: relative ed alla riduzione del margine superiore che la porta verso l’alto. All’inizio è larga il 70% dell’immagine, ma quando si passa sopra col mouse (sia all’immagine che all’etichetta: la classe “thumbnail” è applicata al div padre di entrambe), arriva al 100% in una transizione di un secondo. Quando si toglie il mouse torna alla posizione iniziale nello stesso tempo.

DETTAGLI RISTORANTE

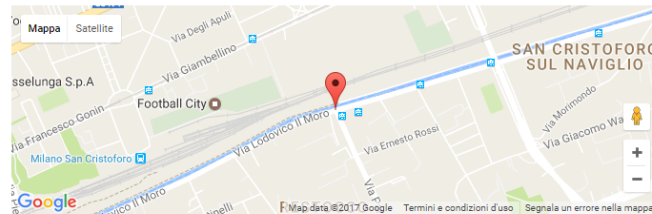
Per accedere a questa schermata è necessario clickare su un ristorante dalla lista dei ristoranti oppure dal risultato della ricerca. Ogni ristorante ha un URL univoco, poiché gli elementi attraverso cui vi si accede sono in realtà dei form che inviano il codice relativo attraverso una GET. Ciò è stato deciso al fine di permettere all’utente di condividere o salvare il link ad un ristorante specifico.

Ogni schermata mostra le informazioni del ristorante relativo, in particolare: nome, indirizzo, città, provincia, tipologia, posizione sulla mappa (ricavata da latitudine e longitudine)

ENTREE

Via Iodovico il moro, 133 - Milano (MI)

Tipologia: cucina mediterranea, cucina tradizionale, vegetariano



Oltre a queste informazioni vengono visualizzati i menù (uno o più per ciascun ristorante), comprensivi di portate (antipasti, primi, secondi, contorni, dessert e bevande, quando presenti), allergeni, prezzo ed eventuali sconti, sia in italiano che in inglese. È possibile prenotare uno specifico menù clickando sul relativo bottone che rimanda alla schermata di prenotazione inizializzando il form con le informazioni del ristorante e menù prescelti.

Vegetariano

Antipasti

flan di zucca
vellutata al taleggio e amaretti

Primi

risotto in crema di asparagi in quartiolo
crema di finocchi d'entrée

Secondi

sformato di radicchio con mirtilli e noci
caprese con fichi

Dessert

trilogia al cioccolato
crema di mascarpone con granella di amaretti
cheesecake con cuore di fragole e pesche sciroppate

Bevande

acqua
vino

allergeni: latte e prodotti a base di latte, sedano e prodotti a base di sedano, lupini e prodotti a base di lupini

prezzo: 25 euro

Nell'orario 19-21 viene applicato lo sconto di 1.00 euro
Nel giorno martedì viene applicato lo sconto di 1.00 euro

Prenota questo menu

Vegetarian

Appetizers

pumpkin flan
taleggio and amaretti cream

First dishes

risotto in asparagus cream with quartiolo
entrée fennel cream

Second dishes

chicory flan with cranberries and walnuts
caprese with figs

Dessert

chocolate trilogy
mascarpone cream with amaretti grains
cheesecake with heart of strawberries and peaches in syrup

Drinks

water
wine

allergens: milk and milk-based products, celery and celery-based products, lupini and lupini-based products

price: 25 euro

At 19-21 is applied the discount of 1.00 euro
On thursday is applied the discount of 1.00 euro

Book this menu

Funzionalità:

- Il contenuto della pagina cambia dinamicamente secondo le informazioni del ristorante (che viene recuperato tramite il codice nell'URL, poiché vi si arriva tramite un form con metodo GET) contenute all'interno dell'array JSON salvato nel Local Storage

```
var nome=(window.location.search).substring(6); //codice passato con la GET
```

Secondo la struttura dell'albero DOM, "window.location" restituisce l'URL della pagina. In particolare "search" restituisce la parte dell'URL relativa ai campi inviati dai form con metodo GET. A questa vengono tolti i primi 6 caratteri per il seguente motivo:

① file:///C:/Users/Teresa/Desktop/Teresa%20Tanzi%20-%20La%20Forkchetta/dettagli_ristorante.html?nome=circolo-lettori

Al click su un ristorante si accede sulla sua pagina specifica, il cui URL è quello sopra. In particolare ci interessa la porzione successiva al "?" che rappresenta il codice inviato dal form nascosto associato alla sua chiave (il "name" del tag input). Rimuovendo i primi 6 caratteri (quindi "?nome=") si riesce quindi ad ottenere il codice vero e proprio.

- I menù in inglese vengono salvati nel JSON proprio come quelli in italiano, ma differenziati tramite una coppia chiave-valore la cui chiave è "lang" ed il valore è "eng" o "ita"

```
"menus": [{
  "lang": "ita",
  "menu": "terra",
  "antipasti": ["salame tipo Montisola", "spiedini di maiale"],
  "primi": ["tagliatelle al salame di Folaga", "tagliatelle con porcini"],
  "secondi": ["manzo all'olio con polenta", "filetto di maiale"],
  "contorni": ["patate fritte", "verdure fresche", "verdure grigliate"],
  "dessert": ["ananas del gallo", "delizie al carrello"],
  "allergeni": ["cereali contenenti glutine", "latte e prodotti a base di latte", "sedano e prodotti a base di sedano", "lupini e"],
  "bevande": ["acqua", "vino bianco del gallo", "vino rosso del gallo"],
  "prezzo": 35.00,
  "sconti": [{
    "tipo": "giorno",
    "valore": ["lunedì", "giovedì"],
    "sconto": 2.00
  }]
}, {
  "lang": "eng",
  "menu": "land",
  "antipasti": ["Montisola sousage", "pork skewers"],
  "primi": ["Folaga sousage noodles", "porcini noodles"],
  "secondi": ["beef in oil with polenta", "pork tenderloin"],
  "contorni": ["french fries", "fresh vegetables", "grilled vegetables"],
  "dessert": ["rooster pineapple", "delights to cart"],
  "allergeni": ["cereals containing gluten", "milk and milk-based products", "celery and celery-based products", "lupini and lupin"],
  "bevande": ["water", "rooster white wine", "rooster red wine"],
  "prezzo": 35.00,
  "sconti": [{
    "tipo": "giorno",
    "valore": ["monday", "thursday"],
    "sconto": 2.00
  }]
}, {
```

Questa porzione di file JSON rappresenta due oggetti dell'array dei menù di un ristorante che, in realtà, rappresentano lo stesso menù, ma in due lingue diverse. Ciò che permette di distinguerli è la chiave "lang".

```

var antipasti=menu["antipasti"];
var antipastiStr="";
if (antipasti.length>0) {
    antipasti.forEach(function (a) {
        antipastiStr+=a+"<br/>";
    })

    var antipastiLabel="";
    if (menu["lang"]=="ita") {
        antipastiLabel="Antipasti";
    } else {
        antipastiLabel="Appetizers";
    }

    appendStr+="

<h5><span class='menu-title'>"+antipastiLabel+"</span></h5>"+antipastiStr+"</p>";
}


```

Per ogni elemento del menù si controlla anzitutto se ha dei valori (ovvero che la lunghezza dell'array superi 0) e, in caso affermativo, la lingua del menù, per poter scrivere l'etichetta relativa nella lingua corretta.

- Al click sul pulsante per la prenotazione vengono salvati nel Session Storage il nome del ristorante e del menù scelto, poi si reindirizza alla pagina per effettuare la prenotazione e si utilizzano queste informazioni per inizializzare i campi del form secondo la scelta effettuata. Terminata l'operazione questa entry viene eliminata dal Session Storage per impedire che, riaccedendo alla pagina tramite un altro percorso, i campi non siano quelli di default

```

$(".btn-prenotazione").click(function() {
    //recupero il menu
    var idMenu=$(this).attr("menu");

    //creo l'oggetto
    var prenotazione=new Object();
    prenotazione.ristorante=nome_ristorante;
    prenotazione.menu=idMenu;

    //salvo le preferenze nel sessionStorage
    window.sessionStorage.setItem('prenotazione', JSON.stringify(prenotazione));

    //mando a prenotazione.html
    window.location.href='prenotazione.html';
})

```

- Il comportamento dei pulsanti per la prenotazione di un determinato menù è lo stesso, sia per quelli in italiano che per quelli in inglese. Solo in questa pagina vengono distinti, in tutte le altre si considerano solo le informazioni in italiano

```

if (menu["lang"]=="ita") {
    //voglio che il tag dei menu sia per entrambi in italiano, questo mi serve perché nella prenotazione
    idMenu=nome;
    appendStr+="

```

Ogni pulsante ha salvato nel suo tag il nome del menù a cui si riferisce: sia che sia in italiano che in inglese viene sempre salvato il nome in italiano.

La variabile “nome” contiene il nome in italiano del menù ricavato dall’array JSON.

RICERCA

Form per la ricerca dei ristoranti secondo diverse caratteristiche, quali: nome, tipologia, città, provincia, fascia di prezzo e numero di posti. Il contenuto dei vari campi verrà poi confrontato con le informazioni di ciascun ristorante presente nel Local Storage. La tipologia viene scelta da un menù a tendina contenente già le varie tipologie di tutti i ristoranti salvati. Con il numero dei posti si intende in generale il numero di posti per la prenotazione legato dalla data e dall’ora di tale, quindi viene confrontato con il numero di posti totale del ristorante. È stata effettuata questa scelta perché l’effettiva disponibilità di posti nella fascia temporale prescelta viene mostrata in fase di prenotazione. Il motivo della ricerca per numero di posti è dovuto a controllare l’effettiva capienza nel caso di gruppi molto grandi.

RICERCA

Nome	<input type="text"/>
Tipologia	<input type="text" value="Qualsiasi"/>
Città	<input type="text"/>
Provincia	<input type="text" value="MI"/>
Fascia di prezzo	<input type="text" value="Qualsiasi"/>
Numero di posti	<input type="text"/>
<input type="button" value="Cerca"/>	

Funzionalità:

- Non tutti i campi sono obbligatori: possono essere compilati solo quelli interessati, gli altri verranno ignorati nella ricerca

Si scorre la lista dei ristoranti salvati. Per ognuno di essi viene salvato un valore booleano in corrispondenza di ciascun campo del form. Questo valore è “true” se il valore inserito nel campo è compatibile con questo ristorante, oppure se in quel campo non viene inserito alcun valore (per far sì che alcuni campi possano essere ignorati). Se non si soddisfa nessuna delle due condizioni la variabile diventa “false”.

```
var isNomeOk=true;
if (nome!=" " && nome.toLowerCase()!=entry["nome"].toLowerCase()) {
    isNomeOk=false;
}
```


Alla fine si controlla se il ristorante è compatibile con tutti i parametri di ricerca e, in caso affermativo, lo si aggiunge alla lista dei risultati (“risultatoRicerca”).

```
if (isNomeOk && isTipoOk && isCittaOk && isProvinciaOk && isPrezzoOk && isPostiOk) {  
    risultatoRicerca.push(entry);  
}
```

- Se non viene compilato alcun campo si intende che qualsiasi ristorante soddisfa la richiesta del cliente, quindi vengono restituiti tutti
- Ogni ristorante che soddisfa la ricerca viene rappresentato sotto al form con un layout simile a quello della pagina contenente la lista di tutti i ristoranti, quando si clicca su uno si visualizza in dettaglio

LA FORCHETTA

HomeRistorantiRicercaPrenotaLogin

Nome

Tipologia

Pizzeria

Citta

Provincia


MI

Fascia di prezzo


Economica

Numero di posti

Cerca



Il Rovescio
Bologna (BO)



Pride Pub
Roma (RM)

- Se nessun ristorante soddisfa la ricerca viene stampato un messaggio al posto dove dovrebbero comparire i ristoranti risultanti. L’assenza di risultati viene dedotta dalla lunghezza dell’array “risultatoRicerca)

RICERCA

Nome	<input type="text"/>
Tipologia	<input type="text" value="Qualsiasi"/>
Città	<input type="text"/>
Provincia	<input type="text" value="MI"/>
Fascia di prezzo	<input type="text" value="Qualsiasi"/>
Numero di posti	<input type="text" value="700"/>
<input type="button" value="Cerca"/>	

La ricerca con questi paramentri non ha dato risultati

```
if (risultatoRicerca.length>0) { //se ho avuto risultati
    risultatoRicerca.forEach(function (entry, index) {
        //se l'indice è pari devo aprire la riga
        if (index%2==0) {
            risultatoStr+<div class="row">'
        }

        //aggiungo il div
        risultatoStr+<div class="col-lg-6 col-md-6"> \
            <form action="dettagli_ristorante.html" method="GET" class="image"> \
                <input type="hidden" name="nome" value="'+entry["codice"]+'"> \
                <button type="submit" class="image-submit"> \
                    <div class="thumbnail ristorante"> \
                         \
                        <div class="wrapper"> \
                            <div class="caption post-content"> \
                                <h3>'+entry["nome"]+'</h3> \
                                <p>'+entry["città"]+' ("'+entry["provincia"]+')</p> \
                            </div> \
                        </div> \
                    </div> \
                </div> \
            </button> \
        </form> \
    </div>';

        //se l'indice è dispari devo chiudere la riga
        if (index%2==1) {
            risultatoStr+</div>'
        }
    })

    $("#risultato-ricerca").html(risultatoStr);
} else { //se non ci sono stati risultati
    $("#risultato-ricerca").append("<div class='center'>La ricerca con questi paramentri non ha dato risultati</div>");
}
```

PRENOTAZIONE

Questa pagina è visualizzabile solo se l'utente è loggato. In caso contrario viene stampato a video un invito ad effettuare l'autenticazione.

PRENOTAZIONE

Per poter prenotare è necessario effettuare il [login](#)

Una volta che l'utente ha effettuato il login è possibile visualizzare il form per prenotare dei posti in un ristorante. Tale richiede le seguenti informazioni: nome del ristorante, nome del menù scelto di quel ristorante, numero di posti nella prenotazione, data e fascia oraria.

PRENOTAZIONE

The screenshot shows a reservation form for 'Antica Trattoria del Gallo'. The 'Ristorante' dropdown is set to 'Antica Trattoria del Gallo', and the 'Menu' dropdown is set to 'Terra'. The 'Numero di posti' field is empty. The 'Data' field is set to '29/03/2017'. The 'Fascia oraria' section has two radio buttons: '19-21' (selected) and '21 in poi'. Below the radio buttons, it says 'Numero posti disponibili: 40'. At the bottom is a dark red 'Prenota' button.

I menù vengono aggiornati automaticamente ogni volta che si cambia il ristorante, in modo tale da permettere di scegliere solo tra i menù offerti da quel ristorante. Il numero di posti prevede un valore minimo pari ad 1 ed un massimo corrispondente alla disponibilità del ristorante nella data e nella fascia oraria selezionata (modificando data ed orario cambia questo massimo, che viene anche indicato in basso per facilitarne la comprensione). È possibile prenotare per una data che parte dal giorno corrente fino ad una settimana in avanti.

PRENOTAZIONE

This screenshot shows the same reservation form as before, but with a validation error. The 'Numero di posti' field now contains the value '41', which is highlighted with a red border. A tooltip message appears next to the field: 'Il valore deve essere inferiore o uguale a 40.' The 'Data' field is set to '30/03/2017', and the 'Fascia oraria' section has '21 in poi' selected. The available seats are still '40'.

PRENOTAZIONE

This screenshot shows a reservation form for 'Soho'. The 'Ristorante' dropdown is set to 'Soho', and the 'Menu' dropdown is set to 'Pescato del giorno'. The 'Numero di posti' field contains the value '2'. The 'Data' field is set to '02/03/2017', and the 'Fascia oraria' section has '21 in poi' selected. A tooltip message appears next to the date field: 'Il valore deve essere 30/03/2017 o successivo.' The available seats are '30'.

Una volta premuto il pulsante per la prenotazione viene mostrato l'importo totale (già comprensivo degli sconti) più alcuni dettagli inerenti la prenotazione, come se fosse una specie di ricevuta.

PRENOTAZIONE

RIASSUNTO PRENOTAZIONE

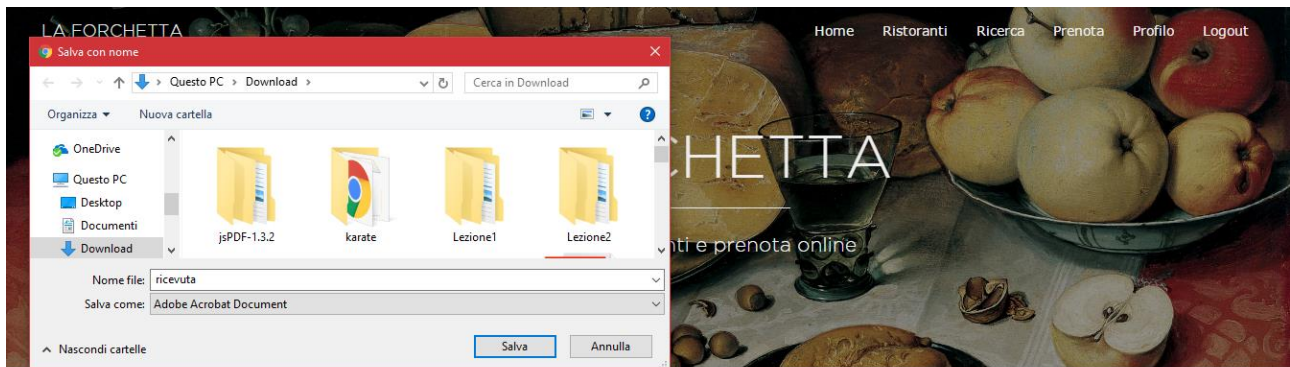
Titolare: Teresa Tanzi
Data dell'ordine: 29/3/2017
Dettagli dell'ordine:

Ristorante	Menu	Data	Fascia oraria	Prezzo	Numero di posti	Importo totale
Aia del Tufo	Tradizionale	2017-03-29	19-21	25.00 €	2	50.00 €

Conferma

Annulla

A questo punto è possibile confermare o annullare la prenotazione: se si decide di annullare si torna al form di prenotazione, altrimenti si reindirizza al profilo (contenente il riassunto di tutte le prenotazioni effettuate dall'utente) e si dà la possibilità di salvare la ricevuta sul proprio pc.



TERESA TANZI

email: teresa.tanzi@studenti.unimi.it

PRENOTAZIONI

Ristorante	Menu	Numero posti	Data	Ora	Importo	Operazioni
Antica Trattoria del Gallo	Terra	2	2017-03-15	19-21	70.00 €	

Funzionalità:

- Si controlla se c'è un utente attivo: per far ciò si cerca nel Session Storage se è presente la coppia con chiave "session". Se non c'è rendo visibile solo il div contenente il link al login

```
//già che accedo alla sessione mi salvo in una variabile l'email dell'utente
var email;
var utente=window.sessionStorage.getItem('session');
if (!utente) {
    //dico di loggarsi
    $("#login-msg").removeClass("hidden").addClass("show");
    //nascondo il form per la prenotazione
    $("#prenotazione-form").removeClass("show").addClass("hidden");
} else {
    $("#login-msg").removeClass("show").addClass("hidden");
    $("#prenotazione-form").removeClass("hidden").addClass("show");
    email=JSON.parse(utente)["email"];
}
```

Questa pagina è strutturata in più div che vengono nascosti e mostrati in base a quali pulsanti l'utente preme o a quali avvenimenti accadono. Di default solo il form per la prenotazione è visibile (classe "show"), ma in questo caso si deve vedere il messaggio che invita al login, quindi gli viene tolta la classe "show" ed aggiunta "hidden", mentre viene fatto il contrario per il messaggio al login.

Se invece l'utente è connesso si fa il contrario e si recupera la sua email.

- Per popolare i menù a tendina si cicla su tutti i ristoranti salvati nel Local Storage (all'inizio non hanno alcuna opzione)

```
//ciclo sui ristoranti e per ciascuno aggiungo il suo nome alla lista dei ristoranti tra cui poter prenotare
listaRistoranti.forEach(function (entry) {
    $("#ristorante").append("<option>" + entry["nome"] + "</option>");
})
```

- Se si arriva dalla selezione di un menù direttamente dalla pagina del ristorante allora imposto come selezionati i valori relativi nei campi del ristorante e del menù. Ricavo tali valori dal Session Storage

```
//controllo se arrivo dalla scelta di un menu in particolare
var preferenze=JSON.parse(window.sessionStorage.getItem('prenotazione'));
if (preferenze) { //se arrivo da delle scelte già fatte le imposto come default
    $("#ristorante option:contains('" + preferenze["ristorante"] + "')").prop("selected",true);

    //ciclo sui menu di questo ristorante, li appendo al select ed imposto quello che arriva come selezionato
    $("#menu").empty();

    var ristoranteSelezionato=$("#ristorante").find(":selected").text();

    listaRistoranti.forEach(function (entry, index) {
        if (ristoranteSelezionato==entry["nome"]) {
            entry["menus"].forEach(function (menu, i) {
                if (i%2==0) {
                    $("#menu").append("<option>" + firstLetterUppercase(menu["menu"]) + "</option>");
                }
            })
        }
    })

    $("#menu option:contains('" + firstLetterUppercase(preferenze["menu"]) + "')").prop("selected",true);

    //elimino la preferenza dal sessionStorage
    window.sessionStorage.removeItem('prenotazione');
} else { //altrimenti imposto come default i menu del primo ristorante in lista
    //setto il default sul menu con quelli del primo ristorante dell'array
    listaRistoranti[0]["menus"].forEach(function (entry, index) {
        if (index%2==0) { //metto solo i menu in italiano (non voglio ripetizioni perché quelli italiani sono più)
            $("#menu").append("<option>" + firstLetterUppercase(entry["menu"]) + "</option>");
        }
    })
}
```

Se nel Session Storage c'è la chiave "prenotazione" impostata dal dettaglio del ristorante si ricava il valore del ristorante e del menù relativo. In particolare i select con id "ristorante" e "menu" già contengono questi valori, è dunque necessario accedere alle opzioni ed individuare quale di queste contiene la stringa precedentemente ricavata. A questo punto si aggiunge a tale opzione la proprietà "selected" impostata a "true".

Prima di andare ad impostare il menù selezionato è necessario riempire il select con le opzioni dei menù offerte dal ristorante. Per far ciò si cerca nella lista dei ristoranti quello selezionato e si prendono tutti i suoi menù di indice pari (quindi tutti quelli in italiano) e vengono appesi al select dei menù.

Se invece si arriva a questa schermata per un'altra via si fa lo stesso procedimento, ma scegliendo come ristorante selezionato il primo della lista e come menù selezionato il primo dei suoi menù.

- Viene calcolato il numero di posti disponibili per il ristorante selezionato, nel giorno e nella fascia oraria scelta

```
function checkPosti(ristorante, data, ora) {  
    //cerco il numero totale di posti del ristorante  
    var listaRistoranti=JSON.parse(window.localStorage.getItem('ristoranti'));  
  
    var totPosti;  
    listaRistoranti.forEach(function (entry) {  
        if (ristorante==entry["nome"]) {  
            totPosti=entry["posti"];  
        }  
    })  
  
    //per ogni prenotazione controllo se è questo ristorante, in questa data ed in quest'ora:  
    var listaPrenotazioni=JSON.parse(window.localStorage.getItem('prenotazioni'));  
    listaPrenotazioni.forEach(function (entry) {  
        if (ristorante==entry["ristorante"] && data==entry["data"] && ora==entry["ora"]) {  
            totPosti-=entry["n_posti"];  
        }  
    })  
  
    return totPosti;  
}
```

Tale funzione prende il numero totale dei posti del ristorante dalla lista dei ristoranti salvata nel Local Storage, poi cicla per tutte le prenotazioni (anch'esse salvate in JSON nel Local Storage) e, per quelle relative allo stesso ristorante, nello stesso giorno ed allo stesso orario, toglie al numero di posti totale il numero di posti riservato per quella specifica prenotazione. In tal modo ottengo il numero dei posti rimanenti che serviranno per valutare se è possibile effettuare la nuova prenotazione

- Le date minima e massima tra cui è possibile scegliere vengono definite dallo script, perché dipendono dalla data corrente (e da essa si calcola la data corrispondente ad una settimana dopo)

```

//imposto il default, il min ed il max dell'imput per la data
var currentDate=new Date();
var day=("0"+currentDate.getDate()).slice(-2);
var month=("0"+(currentDate.getMonth()+1)).slice(-2); //parte a contare i meso da 0, inoltr
var year=currentDate.getFullYear()+1900; //parte a contare gli anni dal 1900
var dateStr=year+"-"+month+"-"+day;

$("#data").attr("value", dateStr);
$("#data").attr("min", dateStr); //la data di oggi è anche la prima data di prenotazione

var nextWeek=new Date(currentDate.getTime()+6*24*60*60*1000); //al tempo di adesso aggiungo
var nextWeekDay=("0"+nextWeek.getDate()).slice(-2);
var nextWeekMonth=("0"+(nextWeek.getMonth()+1)).slice(-2);
var nextWeekYear=nextWeek.getFullYear()+1900;
var nextWeekStr=nextWeekYear+"-"+nextWeekMonth+"-"+nextWeekDay;

$("#data").attr("max", nextWeekStr);

```

Viene recuperata la data corrente dal sistema, ne vengono estratte le componenti e vengono ricomposte in modo da creare una stringa del formato richiesto dal tag dell'imput date. Ai giorni ed ai mesi viene aggiunto 0 all'inizio (in versione stringa) e poi la stringa viene tagliata per tenere solo gli ultimi due elementi. Questo serve perché, per giorni che vanno dal primo del mese al 9 e per mesi che vanno da 1 a 9, si avrebbe un numero di solo una cifra che non viene interpretato correttamente dall'HTML. I mesi partono da 0, è dunque necessario aggiungere 1 al valore ritornato. Gli anni vengono contati dal 1900, bisogna quindi aggiungere questo valore al risultato.

Dopo aver ottenuto la stringa corretta la si utilizza per impostare il valore di default del campo ed anche il suo minimo (non è possibile prenotare per date passate).

Alla data corrente viene aggiunto poi una valore di giorni, ore, minuti, secondi e millisecondi corrispondente ad una settimana: questo permette di ottenere la data del giorno di una settimana avanti che, tramite le stesse modifiche precedenti, viene utilizzato per impostare la data massima dell'input.

- Ogni volta che l'utente cambia la selezione del ristorante viene anche aggiornata la lista dei menù in base al ristorante selezionato

```

//se cambia il ristorante voglio cambiare anche i menu associati
$("#ristorante").change(function() {
    //devo anzitutto ripulire il select dai vecchi risultati
    $("#menu").empty();

    var ristoranteSelezionato=$("#ristorante").find(":selected").text();

    listaRistoranti.forEach(function (entry, index) {
        if (ristoranteSelezionato==entry["nome"]) {
            entry["menus"].forEach(function (menu, i) {
                if (i%2==0) {
                    $("#menu").append("<option>"+firstLetterUppercase(menu["menu"])+ "</option>");
                }
            })
        }
    })
})

```

Quando cambia il valore selezionato nella select dei ristoranti anzitutto si cancellano tutte le opzioni dal select dei menù, poi si scorrono i ristoranti fino a trovare quello selezionato e si appendono le opzioni col nome di tutti i menù di indice pari (quindi in italiano).

- Ogni volta che l'utente cambia ristorante, data o fascia oraria viene aggiornato il numero di posti disponibili

```
//al variare del ristorante, della data o dell'ora modifico posti-disponibili
$("#ristorante").change(function() {
    ristoranteSel=$("#ristorante").find(":selected").text();
    postiRimasti=checkPosti(ristoranteSel, dataSel, oraSel);

    $("#posti-disponibili").html("Numero posti disponibili: "+postiRimasti);
    $("#n-posti").attr("max", postiRimasti);
})
```

Le variabili "ristoranteSel", "dataSel" "oraSel" vengono inizializzate quando la pagina viene caricata in base ai valori di default (rispettivamente: il primo ristorante presente nella lista, la data corrente e la fascia oraria 19-21). Ogni volta che uno di questi valori varia una delle variabili (quella corrispondente) viene sovrascritta e le tre sono utilizzate come parametri per calcolare i posti residui secondo la funzione descritta in precedenza. Il risultato viene impostato come massimo dell'input del numero di posti. Inoltre viene visualizzato come testo in un div per mostrare a video il massimo di posti che si possono selezionare per una prenotazione.

- Il calcolo dell'importo totale viene fatto in base al prezzo del menù a cui vengono sottratti gli sconti in base al giorno ed alla fascia oraria selezionata, il risultato viene poi moltiplicato per il numero di posti nella prenotazione


```

//scorro la lista dei menu del ristorante per trovare quello giusto
entry["menus"].forEach(function (m) {
  if (menu.toLowerCase()==m["menu"]) {
    //salvo il prezzo del menu
    prezzo=m["prezzo"];

    //devo ora applicare gli sconti relativi a questo menu, al giorno e all'ora
    if (m["sconti"]>0) {
      m["sconti"].forEach(function (s) {
        if (s["tipo"]=="giorno") {
          var scontoGiorno=false;
          s["valore"].forEach(function (value) {
            var myDate=new Date (data); //metto l'input ricevuto in formato date per p
            if (value=="lunedì" && myDate.getDay()==1 || value=="martedì" && myDate.getDay()==2) {
              scontoGiorno=true;
            }
          })
          if (scontoGiorno) {
            prezzo-=s["sconto"];
          }
        }

        if (s["tipo"]=="orario") {
          var scontoOrario=false;
          s["valore"].forEach(function (value) {
            if (value==ora) {
              scontoOrario=true;
            }
          })
          if (scontoOrario) {
            prezzo-=s["sconto"];
          }
        }
      })
    }
  })
})
}
})

```

Anzitutto si salva il prezzo del menù in una variabile. Si scorre poi l'array degli sconti (se contenente un elemento o più): esistono due tipi di sconti, quelli che si applicano al giorno e quelli che si applicano all'orario. Un particolare menù può non avere alcun tipo di sconto, può avere lo sconto per un solo tipo o averne per entrambi. Ogni tipologia sottrae quindi lo sconto separatamente. Per quanto riguarda sconti sui giorni si controlla se il giorno della settimana del giorno selezionato (un intero compreso tra 1 e 7) corrisponde al valore della stringa salvata nel JSON. In particolare 1 corrisponde a "lunedì", 2 a "martedì" e così via. Per l'orario è più comodo, perché il valore dei radio button corrisponde a quello degli sconti. Infine la variabile "prezzo" viene sovrascritta, per entrambi gli conti, con se stessa meno il valore dello sconto, così che alla fine contenga il prezzo effettivo (che va poi moltiplicato per il numero di posti).

- Alla conferma della prenotazione i dati relativi ad essa vengono aggiunti all'array JSON delle prenotazioni all'interno del Local Storage. Se questo ancora non esiste (perché ancora non sono state effettuate prenotazioni) viene prima creato
- Una volta che la prenotazione viene effettuata viene aperto un popup per permettere di salvare il PDF inerente la prenotazione, come una sorta di ricevuta

```
//stampo il pdf della prenotazione
var doc=new jsPDF('l', 'pt', 'a4');

doc.fromHTML($('#canvas').get(0), 15, 15, {
  'width': 170
});

doc.save('ricevuta.pdf');
```

Per fare questa funzione è stata utilizzata la libreria “jsPDF”. Viene anzitutto creato un documento PDF vuoto orientato in landscape, di formato A4 e le cui misure vengono fatte in punti. Il documento viene poi riempito con la visualizzazione del contenuto del tag con id “canvas” (per accedere al contenuto si usa la funzione get(0)) contenente la tabella riassuntiva della prenotazione, tuttavia ne viene ignorato lo stile. Gli altri valori indicano i margini e la larghezza. Alla fine il documento viene salvato col nome “ricevuta.pdf”.

Per far sì che venisse salvato anche lo stile sarebbe stato necessario fare uno screenshot alla porzione di schermo contenente “canvas” (grazie alla libreria “html2canvas”) ed aggiungere quell’immagine al documento, tuttavia il risultato viene piuttosto sfuocato, pertanto si è deciso di mantenere la prima soluzione, più povera di stile, ma meglio visualizzata.

PROFILO

Questa pagina è visibile solo se l’utente è al momento connesso. In caso contrario nella navbar non è presente il link per accedervi.

Qui vengono mostrate le informazioni relative all’utente attivo, in particolare: nome, cognome, email e la lista di tutte le prenotazioni effettuate da tale utente. Da questa lista è possibile eliminare le prenotazioni la cui data ancora non è passata (ad esclusione del giorno corrente, per impedire di disdire all’ultimo momento).

TERESA TANZI

email: teresa.tanzi@studenti.unimi.it

PRENOTAZIONI

Ristorante	Menu	Numero posti	Data	Ora	Importo	Operazioni
Aia del Tufo	Tradizionale	4	2017-03-17	21 in poi	100.00 €	
Entree	Vegetariano	2	2017-03-23	19-21	48.00 €	
Circolo dei Lettori	Degustazione piemontese	1	2017-03-27	19-21	68.00 €	
Soho	Pescato del giorno	2	2017-03-23	21 in poi	50.00 €	
Pride Pub	Pizze a scelta	2	2017-03-23	19-21	26.00 €	
Soho	Insalatone	3	2017-04-02	21 in poi	39.00 €	Elimina
Entree	Vegetariano	2	2017-03-31	19-21	48.00 €	Elimina
Aia del Tufo	Tradizionale	2	2017-04-02	21 in poi	50.00 €	Elimina

Modifica profilo Elimina account

È possibile modificare le informazioni che vengono date in fase di registrazione, eccezion fatta per l’email che viene considerata come identificativo dell’utente e per questo associata a molteplici informazioni. Il form addetto alla modifica del profilo compare in fondo alla pagina nel momento in cui viene premuto il pulsante relativo.

Modifica profilo

Elimina account

Nome

nome

Cognome

cognome

Password

Modifica

Annulla

Quando si conferma una modifica la pagina viene ricaricata per aggiornare il profilo con le nuove informazioni, tuttavia c'è un caso in cui è impossibile avere un feedback dell'avvenuta modifica in questo modo, ovvero nel caso in cui si cambi solo la password (questo perché la password non viene mostrata). Per risolvere questo problema viene mostrata una snackbar temporanea, simile a quella tipica di Android, in fondo allo schermo con un messaggio di avvenuta modifica.



TERESA TANZI

email: teresa.tanzi@studenti.unimi.it

PRENOTAZIONI

Le informazioni sono state modificate con successo

È inoltre possibile eliminare il proprio account clickando sull'apposito pulsante. Tuttavia, prima di avviare la cancellazione definitiva, viene richiesta una conferma ulteriore per prevenire eventuali errori.

Modifica profilo

Elimina account

Sei sicuro di voler eliminare il tuo profilo?

Conferma

Annulla

Una volta confermata la cancellazione si viene reindirizzati alla homepage e, anche in questo caso, si mostra una snackbar per confermare l'operazione. Le informazioni relative all'account vengono eliminate, ma non quelle relative alle prenotazioni (questo perché ormai le prenotazioni sono state fatte e sono ancora valide).



Stai cercando un ristorante?

Il filosofo Ludwig Feuerbach scriveva che "noi siamo ciò che mangiamo"... Ciò che mangiamo rifletterebbe chi siamo, chi diventeremo, ma anche i nostri principi etici. Esiste un'etica nell'alimentazione, una propensione naturale nel cercare nel cibo la pace e l'armonia che vogliamo per la natura, l'umanità, la nostra salute e per noi stessi, poiché ogni nostra scelta ha sempre le sue conseguenze.

La filosofia de LA FORCHETTA è quella di offrire un'esperienza unica nello spazio, ma anche nel gusto, selezionando con cura le proposte al fine di valorizzare e far conoscere quelle attività che si impegnano nella cura delle persone e dell'ambiente con la passione di chi sa regalare i prodotti migliori.

L'account è stato eliminato con successo

Funzionalità:

- Dal Session Storage viene recuperata l'email dell'utente attivo e viene confrontata con le informazioni degli utenti nel Local Storage per trovare e stampare nome e cognome e la lista delle prenotazioni

```
//recupero l'utente dal local storage confrontando l'indirizzo in sessione con gli utenti salvati
var utenteAttivo=window.sessionStorage.getItem("session"); //string
var email=JSON.parse(utenteAttivo)["email"]; //email
var listaUtenti=window.localStorage.getItem('utenti'); //string
var listaJson=JSON.parse(listaUtenti); //JSON

listaJson.forEach(function(entry) {
    if (email==entry["email"]) {
        console.log(JSON.stringify(entry));
        $("#nome_utente").html(entry["nome"]+" "+entry["cognome"]);

        $("#dettagli_utente").html("email: "+entry["email"]);
    }
})
```

```

//popolo la lista delle prenotazioni
var prenotazioni=[];
var listaPrenotazioni=JSON.parse(window.localStorage.getItem('prenotazioni'));
listaPrenotazioni.forEach(function (entry) {
    if (entry["utente"]==email) {
        prenotazioni.push(entry);
    }
})

var prenotazioniStr="";
if (prenotazioni.length>0) {
    prenotazioniStr+="

#### <b>PRENOTAZIONI</b></h4><table class='table'><thead><t"; prenotazioni.forEach(function (entry) { prenotazioniStr+="


```

La lista delle prenotazioni viene creata aggiungendo ad un array tutte le prenotazioni che hanno nel campo “utente” l’email dell’utente attivo. Se effettivamente ci sono delle prenotazioni a suo nome viene creata una tabella contenete tutte le informazioni salvate nella prenotazione. Si controlla poi la data della prenotazione e, se è successiva alla data corrente (per fare il confronto è necessario convertire la stringa salvata della data della prenotazione in un oggetto Date), viene aggiunto il pulsante per l’eliminazione.

- Modifica delle informazioni personali quali: nome, cognome e password attraverso un form. È possibile cambiare una o più di queste contemporaneamente

```

if (password!="") {
    //per cambiare il dato basta sovrascriverlo
    listaJson.forEach(function(entry) {
        if (email==entry["email"]) {
            entry["password"]=password;
        }
    })
    window.sessionStorage.setItem('modifica', 'true');
}

window.localStorage.setItem('utenti', JSON.stringify(listaJson));

```

Per ogni elemento del form si controlla se è stato inserito qualcosa (altrimenti si sovrascriverebbe le informazioni con una stringa vuota). listaJson è la lista degli utenti: viene scorsa per trovare l'utente attivo e reimpostare il campo modificato.

Per ogni modifica si salva nel Session Storage un flag che indica che la modifica è avvenuta: serve per avviare il messaggio di feedback.

Quando sono state aggiunti tutti i campi modificati, l'array degli utenti viene ricaricato nel Local Storage. La pagina viene in seguito ricaricata per mostrare le informazioni corrette. Per far ciò è stato sufficiente impostare, come azione del form, la pagina stessa.

- Eliminazione del profilo dalla lista degli account (chiedendo una conferma, data la sensibilità dell'operazione). Tuttavia le prenotazioni effettuate da tale utente vengono mantenute, questo perché si intende la prenotazione come un impegno esterno al sito, pari ad una prenotazione telefonica diretta con un ristorante

```
//se confermo la cancellazione allora elimino definitivamente il profilo da quelli salvati
$("#conferma-cancellazione").click(function() {
    //elimino l'entry dall'array degli utenti e lo ricarico
    listaJson.forEach(function(entry, index) {
        if (email==entry["email"]) {
            //elimino un elemento
            listaJson.splice(index, 1); //parte dall'indice, va avanti di uno e, nell'intervallo formato dal mio unico elem
        }
        console.log(JSON.stringify(listaJson));
    })
    window.localStorage.setItem('utenti', JSON.stringify(listaJson));

    //cancello l'utente dallo storage
    window.sessionStorage.removeItem('session');

    //salvo nello storage che è avvenuta una cancellazione, così quando mando all'index mando la conferma di avvenuta cancellazione
    window.sessionStorage.setItem('cancellazione', 'true');

    //modifico la navbar
    $("#navbar").children("li").eq(5).remove();
    $("#navbar").children("li").eq(4).remove();
    $("#navbar").append('<li><a href="profilo.html">Profilo</a></li><li><a href="index.html" id="logout">Logout</a></li>');

    //imposto come azione del form l'index
    $('#cancella-profilo').attr('action', 'index.html');
    //window.location.replace("index.html");
})
```

Si scorre la lista degli utenti e, quando si trova l'utente attivo, lo si rimuove. La lista degli utenti viene quindi aggiornata sul Local Storage, si rimuove dal Session Storage la variabile che indica l'utente attivo (non avendo più un account viene disconnesso e non può più accedere alle pagine riservate come il profilo) e viene salvato un flag che indica che è avvenuta una cancellazione (per mostrare a video il messaggio di conferma in un'altra schermata).

Non essendo più loggato è necessario cambiare la navbar togliendo i link al profilo ed al logout, aggiungendo però quello al login.

- È possibile cancellare le prenotazioni per date dal giorno seguente in poi.


```

//listener per i pulsanti di cancellazione su ciascuna riga della tabella delle prenotazioni
$(".table-btn").click(function() {
    //recupero le informazioni della prenotazione dalla tabella
    var ristorante_c=$(this).closest('tr').find('td:eq(0)').text(); //closest() trova l'elemento più
    //find() trova invece l'elemento scendendo nell'albero
    //:eq() seleziona l'elemento all'indice indicato
    var menu_c=$(this).closest('tr').find('td:eq(1)').text();
    var posti_c=$(this).closest('tr').find('td:eq(2)').text();
    var data_c=$(this).closest('tr').find('td:eq(3)').text();
    var ora_c=$(this).closest('tr').find('td:eq(4)').text();

    //recupero l'oggetto delle prenotazioni dal local storage
    var prenotazioniArray=JSON.parse(window.localStorage.getItem('prenotazioni'));

    //scorro l'array delle prenotazioni e trovo quella con le caratteristiche di quella selezionata
    prenotazioniArray.forEach(function (entry, index) {
        if (entry["utente"]==email && entry["ristorante"]==ristorante_c && entry["menu"]==menu_c &&
            //elimino la entry dall'array delle prenotazioni
            prenotazioniArray.splice(index, 1);
            //ricarico l'array nello storage
            window.localStorage.setItem('prenotazioni', JSON.stringify(prenotazioniArray));
            //ricarico la pagina
            window.location.reload();
        }
    })
})

```

Le informazioni relative alla prenotazione da cancellare sono recuperate percorrendo l'albero DOM sulla tabella (dal pulsante – elemento di una cella – si sale alla riga a cui appartiene e da questa si scende verso varie celle, numerate secondo la posizione). Si scorre poi la lista delle prenotazioni e si trova quella con gli stessi campi di quelli recuperati dalla tabella per poter poi eliminare quella corretta. Alla fine la pagina viene ricaricata per non mostrare più tale prenotazione.

LOGIN

Questa pagina è visibile solo per utenti che ancora non si sono autenticati. In caso contrario al suo posto nella navbar compare il pulsante per effettuare il logout.

Permette sia ad utenti in possesso di un account di effettuare il login, sia agli altri di creare un nuovo account. I due form coesistono nella stessa schermata ed indirizzano l'utente verso quello interessato tramite una breve stringa esplicativa. Le informazioni richieste per registrazione sono nome, cognome, email e password, mentre per autenticarsi servono email e password.

Sei nuovo? Allora iscriviti:

Email

Nome

Cognome

Password

Hai già un account? Accedi:

Email

Password

Per quanto riguarda la registrazione si controlla che l'email inserita non corrisponda ad un account già esistente. Per il login si controlla invece che l'email esista tra quelle salvate nella lista degli utenti e che la password sia quella corrispondente. Tutti i campi devono essere riempiti e si controlla che l'email sia nel formato corretto.

The image shows two side-by-side screenshots of a web application's user interface. The left screenshot is for registration, titled 'Sei nuovo? Allora iscriviti:'. It contains four input fields: 'Email' (filled with 'teresa.tanzi@studenti.unimi.it'), 'Nome' (filled with 'Teresa'), 'Cognome' (filled with 'Tanzi'), and 'Password' (filled with three dots). A red error message at the top says 'Impossibile creare un nuovo account: questa email è già associata ad un account esistente'. A dark red 'Iscriviti' button is at the bottom. The right screenshot is for login, titled 'Hai già un account? Accedi:'. It contains two input fields: 'Email' (filled with 'teresa.tanzi|') and 'Password' (filled with three dots). A red error message at the top says 'Impossibile effettuare il login: la password non è corretta'. A red 'Accedi' button is below the password field. A tooltip points to the password field, stating: 'Aggiungi un simbolo "@" nell'indirizzo email. In "teresa.tanzi" manca un simbolo "@".'.

Entrambi i form, nel caso in cui le informazioni inserite siano corrette, reindirigono al profilo dell'utente.

Funzionalità:

- Quando si effettua la registrazione si controlla se già esiste un array nel Local Storage per contenere la lista degli utenti in possesso di un account. In caso contrario questo array viene creato vuoto. Si procede poi all'inserimento dei nuovi valori nell'array (ovviamente se superano il controllo)

```
//controllo nello storage se c'è un array per gli utenti: se non c'è lo creo, altrimenti nulla
var listaUtenti=window.localStorage.getItem('utenti');
var utentiJson;
if (listaUtenti) {
  //esiste già l'array: lo recupero
  utentiJson=JSON.parse(listaUtenti); //array di utenti
  console.log("utenti: "+JSON.stringify(utentiJson));
} else {
  //creo l'array vuoto (tanto di utenti non ce n'erano)
  utentiJson=[];
}
```

- Controllo dell'esistenza di un account con la stessa email per la registrazione: se esiste non si crea un nuovo account e si stampa un errore


```

//controllo se l'utente ha già un account
var noAccount=true;
$.each(utentiJson, function (index, entry) {
    var storedEmail=entry["email"];

    if (email==storedEmail) { //l'utente ha già un account: non può registrarsi
        //alert ("Hai già un account");
        $("#errore-registrazione").html("Impossibile creare un nuovo account: questa email è già");
        noAccount=false;
        return false;
    }
})

if(noAccount) {
    //aggiungo il nuovo utente all'array di utenti
    utentiJson.push(newUtente);
    success ("#registrazione", email);
    //salvo l'array degli utenti nello storage al posto di quello di prima
    window.localStorage.setItem('utenti', JSON.stringify(utentiJson));
} else {
    return false; //non ricarico la pagina
}
}

```

La variabile booleano “noAccount” vale true se non esiste alcun account associato a quella email. Si scorre quindi la lista degli utenti e, quando se ne trova uno con la stessa email di quella inserita, “noAccount” cambia valore e diventa false. Viene inoltre mostrato un messaggio che indica l’impossibilità di creare un account con la stessa email di uno esistente. Se invece l’account non esiste ancora (“noAccount” vale true), viene generato aggiungendo le informazioni collezionate tramite form nella lista. L’utente viene poi collegato tramite la funzione “success”.

- Controllo dell’esistenza di un account con la stessa email per l’autenticazione e della corrispondenza della password: se non vanno bene viene stampato un messaggio di errore

```

//ciclo sull'array e controllo che esista l'utente e che la password sia corretta
var nonRegistrato=true;
$.each(utentiJson, function (index, entry) {
    var storedEmail=entry["email"];
    var storedPassword=entry["password"];

    if (email==storedEmail) { //l'utente è iscritto
        if (password==storedPassword) { //la password inserita è quella corretta
            success ("#login", email);
            nonRegistrato=false;
            return false;
        } else { //la password inserita non è quella associata a tale utente
            //alert ("Wrong password");
            $("#errore-login").html("Impossibile effettuare il login: la password non è corretta");
            nonRegistrato=false;
            return false;
        }
    }
})

if (nonRegistrato) {
    //alert ("Utente non registrato");
    $("#errore-login").html("Impossibile effettuare il login: non esiste un account associato a questa email");
    return false;
}
}

```

Analogamente alla funzione precedente, viene inizializzata una variabile booleana a true per indicare che non esiste un account associato alla email inviata nel form. Si cicla poi su tutti gli account e, quando se ne trova uno con la stessa email, la variabile diventa false. Se resta vera anche dopo al termine del ciclo viene visualizzato un messaggio d'errore. Se invece viene trovato l'account associato si passa al controllo della password: se non coincide con quella salvata viene inviato un altro messaggio d'errore, altrimenti è tutto in regola per effettuare l'accesso tramite la funzione "success".

- Quando si accede, sia ad un account nuovo che ad uno esistente, viene salvata l'email nel Session Storage per indicare quale sia l'utente attivo e si rimanda al profilo personale. La modifica della navbar (per aggiungere le aree accedibili solo da loggati), viene fatta nella pagina del profilo.

```
//se la registrazione o il login sono andati a buon fine allora avvio la sessione  
function success(formId, email) {  
  //salvo nello storage l'email dell'utente che è ora attivo  
  var utenteLog=new Object;  
  utenteLog.email=email;  
  window.sessionStorage.setItem('session', JSON.stringify(utenteLog));  
  
  //redirect a profilo.html  
  $(formId).attr('action', 'profilo.html');  
}
```

Alla funzione viene passato l'id della form e l'email dell'utente che si deve loggare. L'email viene aggiunta al Session Storage per segnalare alle altre pagine l'utente attivo, mentre l'id della form serve per modificare l'azione di quella in modo tale che rimandi al profilo.

Ogni schermata è composta da 4 elementi:

- Navbar: cambia dinamicamente in base alla presenza o meno di un utente attivo. All'inizio, quando non è ancora connesso un account, presenta un pulsante fisso con etichetta "LA FORCHETTA" che rimanda alla homepage, poi i link alle pagine Home, Ristoranti, Ricerca, Prenota, Login.



Quando invece un utente si autentica i pulsanti cambiano e diventano Home, Ristoranti, Ricerca, Prenota, Profilo, Logout.



Di default le pagine mostrano la prima versione, quella per gli utenti generici. Viene poi eseguita la funzione che controlla se esiste un utente connesso: in caso affermativo prende la navbar ed elimina il suo figlio (quindi un list item) a posizione 4, ovvero "Login". A questa vengono poi appesi altri due list item: "Profilo", che rimanda alla pagina del profilo, e "Logout", diretto alla homepage. In caso contrario si esegue lo stesso tipo di operazioni per ottenere la versione della navbar per gli utenti non collegati.

```
function session() {
  //controlla nello storage se c'è un utente loggato
  var sessione=window.sessionStorage.getItem('session');
  if (sessione) { //un utente è loggato
    //cambio la barra di navigazione per poter vedere il profilo, aggiungo il logout ed elimino il login
    $("#navbar").children("li").eq(4).remove();
    $("#navbar").append('<li><a href="profilo.html">Profilo</a></li><li><a href="index.html" id="logout">Logout</a></li>');
  } else {
    //modifico la navbar
    $("#navbar").children("li").eq(5).remove();
    $("#navbar").children("li").eq(4).remove();
    $("#navbar").append('<li><a href="login.html">Login</a></li>');
  }
}
```

Il colore di sfondo della barra cambia in base alla posizione della barra per scrollare la pagina: se si è scesi di un valore minore ai 10 pixel è trasparente e si integra con l'immagine del banner, altrimenti si colora di bordeaux per rimanere ben visibile in qualsiasi posizione della pagina (infatti è fissa al top della pagina)



```
//quando scrollo la pagina cambio il colore alla navbar
var scroll_pos=0;

$(document).scroll(function() {
  scroll_pos=$(this).scrollTop();

  if (scroll_pos>10) {
    console.log("yess: "+scroll_pos);
    //$("#heder").css('background-color', 'red');
    $(".navbar-fixed-top").removeClass("on-top").addClass('not-top');
  } else {
    console.log("nop "+scroll_pos);
    $(".navbar-fixed-top").removeClass('not-top').addClass('on-top');
  }
});
```

Appena la pagina viene caricata la variabile “scroll_pos”, che indica la distanza dal top della pagina, viene inizializzata a 0 perché nella schermata viene visualizzata la parte più alta della pagina. Ogni volta che la pagina viene scrollata, “scroll_top” viene assegnata con il numero di pixel che sono nascosti al di sopra dell’area visualizzata. Se “scroll_top” è maggiore di 10, quindi si è scesi un pochino nella pagina, alla navbar viene rimossa la classe “on-top” e viene aggiunta “not-top”, viceversa nel caso contrario.

```
.on-top {
  background-color: transparent;
}

.not-top {
  background-color: #501a1a;
}

.navbar-fixed-top {
  transition-duration: 1s;
}
```

Queste due classi si occupano di cambiare il colore di sfondo della navbar. Inoltre si imposta un valore per la durata delle trasformazioni inerenti la navbar, nello specifico 1 secondo, per evitare una transizione netta.

- Banner: grande immagine iniziale su cui vengono stampati il nome dell'applicazione e lo slogan, oltre alla barra di navigazione (che in seguito, scrollando la pagina, si sposta). È esterno al container per coprire tutto lo spazio possibile sia ai lati che in alto. A differenza della navbar però non è fissa, quindi scompare man mano mentre si scende nella pagina



- Container: elemento che cambia per ogni pagina e contiene il layout relativo alle funzionalità di ciascuna di esse
- Footer: piccola barra colorata che indica la terminazione della pagina. Riporta alcune informazioni inerenti il progetto (come il proprietario) ed il link ai principali social network cui l'applicazione potrebbe essere collegata



Le principali strutture dati sono degli array JSON e vengono salvate all'interno del Local Storage. Queste rappresentano informazioni riguardo:

- Ristoranti: per ogni ristorante viene memorizzato un codice (identificativo che viene mostrato nel URL della pagina), il nome, una foto, la lista delle tipologie, l'indirizzo (via, città, provincia e coordinate GPS), la fascia di prezzo, il numero di posti e la lista dei menù. Ogni menù contiene informazioni quali il nome, la lingua in cui è descritto, le portate (array di antipasti, primi, secondi, contorni, dessert e bevande), il prezzo ed un array per gli sconti. Ogni oggetto sconto ha un tipo (elemento temporale a cui viene applicato), un valore (in quale istanze temporali di quel tipo viene applicato) ed il valore vero e proprio dello sconto
- Utenti: per ogni utente vengono salvate le informazioni che vengono date all'iscrizione, ovvero email, nome, cognome e password
- Prenotazioni: per ogni prenotazione si conoscono l'email dell'utente che l'ha effettuata, il ristorante ed il menù scelti, il numero di posti, il giorno, la fascia oraria e l'importo totale

Non si è ritenuto necessario creare una struttura dati per memorizzare lo stato di ciascun ristorante per ogni fascia temporale, poiché ricavabile già dalle informazioni contenute nella lista dei ristoranti e delle prenotazioni.