Progetto di Applicazioni Web e Cloud – Applicazione Web

Roberto Pinotti roberto.pinotti@studenti.unimi.it

1) Analisi dei requisiti

L'obiettivo del progetto è sviluppare un'applicazione web simile a www.thefork.com, cioè un sistema per la ricerca e la prenotazione di ristoranti.

2) Identificazione delle funzionalità da sviluppare

Il sistema è composto da dua macro scenari:

- La gestione dei ristoranti giorno per giorno

I ristoranti sono gestiti in finestre temporali di una settimana (da lunedì a domenica) e contengono l'associazione tra giorno, ora e prenotazione. L'utente, se registrato, ha la possibilità di prenotare un tavolo scegliendo ristorante, data e ora.

Ogni ristorante possiede le seguenti informazioni: nome, tipo di cucina, luogo, prezzo, stelle, eventuali sconti, menù. Il menù, caratterizzato da un nome, contiene le portate (antipasti, primi, secondi), gli allergenici, le bevande e il prezzo. Tutti questi dati possono essere visionati anche da utenti non registrati.

È possibile ricercare i ristoranti in base nome, luogo, tipo di cucina e prezzo.

- La gestione delle prenotazioni di utenti registrati

Gi utenti che vogliono prenotare un tavolo si devono registrare.

Una volta registrato, o dopo avere fatto login, un utente può prenotare tutti i posti che vuole in tutti i ristoranti che vuole. Le fasce orarie di prenotazione sono due: dalle 19 alle 21 e dopo le 21. Il totale dei posti prenotabili in queste due fasce dipende dalla disponibilità residua dei posti nel singolo ristorante.

Ogni utente è caratterizzato da un nome e cognome e da una password. Queste due informazioni possono essere modificate a piacere.

3) Progettazione della struttura e della presentazione delle pagine web

Per la creazione dei sito web mi sono servito del framework Bootstrap (www.getbootstrap.com). Ho scelto Bootstrap perché in poche righe di codice permette di creare siti responsive (il traffico web da mobile è circa il 60% del traffico web totale, fare dei siti mobile-first è indispensabile).

La struttura del progetto - e di Bootstrap - è la seguente:

```
the-fork/
├── css/
│   ├── bootstrap.css
│   └── bootstrap-united.css
├── js/
│   ├── vendor/
│   │       ├── bootstrap.js
│   │       ├── bootstrap.min.js
│   │       ├── jquery-1.11.2.js
│   │       ├── modernizr-2.8.3-respond-1.4.2.min.js
│   │       ├── npm.js
│   │       └── plugins.js
│   └── ristoranti.json
├── fonts/
│   ├── glyphicons-halflings-regular.eot
│   ├── glyphicons-halflings-regular.svg
│   ├── glyphicons-halflings-regular.ttf
│   └── glyphicons-halflings-regular.woff
└── pagine html
```

I file `bootstrap.css` e `bootstrap-united.css` contengono la parte grafica del sito web.
Il file `ristoranti.json` contiene le informazioni dei ristoranti in formato JSON.
Dentro la cartella `vendor/` sono presenti i file javascript di Bootstrap.
Nella cartella `/fonts` sono presenti le icone di Bootstrap utlizzabili nel progetto.

Le pagine html sono:
- `index.html`: è la home e contiene l'elenco di tutti i ristornati nel database e la funzione di ricerca;
- `prenotazione.html`: contiene le informazioni del singolo ristorante e permette ad un utente registrato e loggato di prenotare un tavolo;
- `registrazione.html`: contiene il form per la registrazione al servizio;
- `entra.html`: contiene il form per il login al servizio;
- `account.html`: contiene le informazioni dell'utente e le prenotazione effettuate;
- `modificanomecognome.html` e `modificapassword.html` contengono i form per aggiornare rispettivamente il nome e cognome e la password

La cartella `/en/` contiene le pagine per il sito in lingua inglese.

4) Progettazione della sorgente di informazioni statica o dinamica

Le strutture dati presenti nel progetto sono due:

- I ristoranti

Le informazioni del ristoranti sono contenute in un file JSON. La struttura del file JSON è molto semplice: i ristoranti sono in un unico grande vettore e ogni ristorante, a sua volta, ha un vettore `menu` e un vettore `prenotazione` per gestire rispettivamente i menu disponibili e le prenotazione degli utenti.

Questi dati vengono salvati nel localStorage del browser al primo avvio del sito web. A questo punto i dati diventano dinamici: saranno modificati nel localStorage, non nel file JSON, in base alle azioni dell'utente (ad esempio la prenotazione di un tavolo).

La struttura dati presente nel localStorage rispecchia perfettamente quella del JSON.

Esempio: `[{"id":0,"nome":"Roberto Pizza","tipo":["Pizzeria"],"luogo":"Via Pizzo Camino,
Dalmine","prezzo":"€€","stelle":4.1,"foto":"foto/0.jpg","menu":[{"nome":"Classico","antipasti":[""],"primi":["Pizza semplice a scelta"],"secondi":[""],"allergia":["Latte","Uovo","Frumento"],"bevande":["Chinotto in lattina","Aranciata in lattina","Limonata in lattina"],"prezzo":6},{"nome":"Medio","antipasti":[""],"primi":["Pizza farcita a scelta"],"secondi":[""],"allergia":["Latte","Uovo","Frumento"],"bevande":["Chinotto da 0.5L","Aranciata da 0.5L","Limonata da 0.5L"],"prezzo":8},{"nome":"Maxi","antipasti":[""],"primi":["Pizza spaciale a scelta"],"secondi":[""],"allergia":["Latte","Uovo","Frumento"],"bevande":["Chinotto da 0.5L","Aranciata da 0.5L","Limonata da 0.5L"],"prezzo":10}],"sconto":{"tipo":"Studenti","valore":"20"},"prenotazione":[{"giorno":"Lunedì","tavolo":[{"orario":"19-21","posti":18},{"orario":"21+","posti":20}]},{"giorno":"Martedì","tavolo":[{"orario":"19-21","posti":20},{"orario":"21+","posti":20}]},{"giorno":"Mercoledì","tavolo":[{"orario":"19-21","posti":20},{"orario":"21+","posti":20}]},{"giorno":"Giovedì","tavolo":[{"orario":"19-21","posti":20},{"orario":"21+","posti":20}]},{"giorno":"Venerdì","tavolo":[{"orario":"19-21","posti":20},{"orario":"21+","posti":20}]},{"giorno":"Sabato","tavolo":[{"orario":"19-21","posti":20},{"orario":"21+","posti":20}]},{"giorno":"Domenica","tavolo":[{"orario":"19-21","posti":20},{"orario":"21+","posti":20}]}]
}, …]`

- Gli utenti

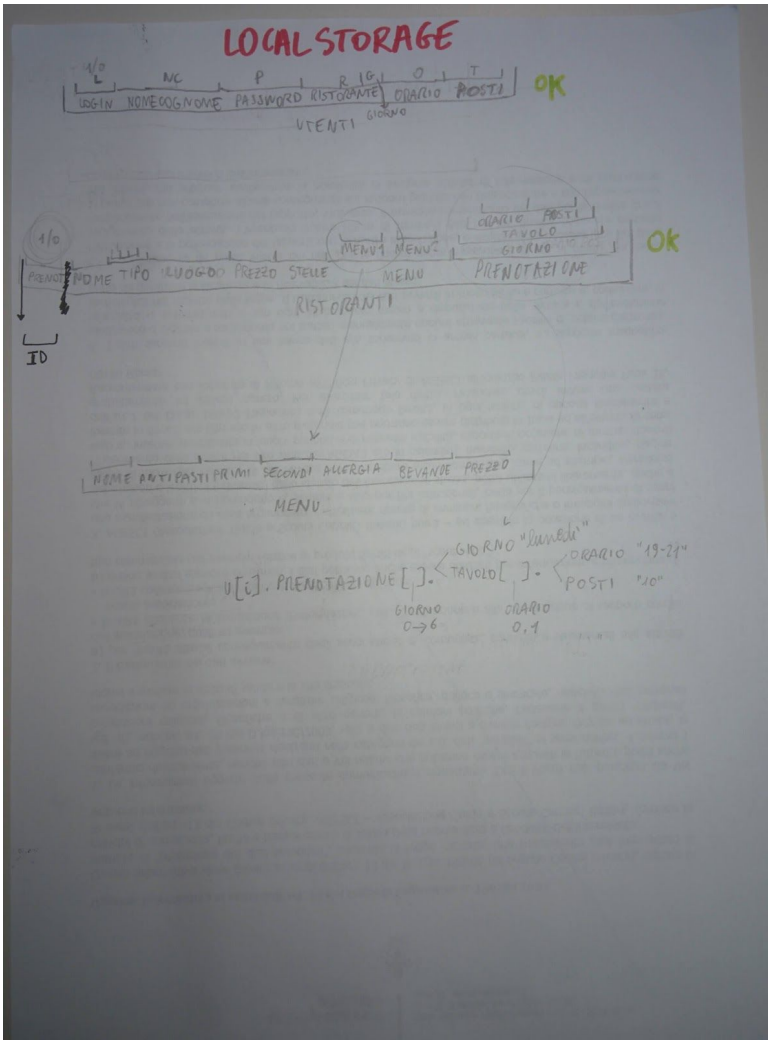
L'altra strattura dati presente nel sistema è quella per la gestione degli utenti registrati. Questa struttura non viene creara a partire da un file JSON ma viene creata (vuota) direttamente nel localStorage al primo avvio del sito web. La struttura si modificherà dinamicamente in base alle azioni dell'utente (ad esempio la registrazione di un nuovo utente o la prenotazione di un tavolo).

La struttura dati presente nel localStorage è fatta così: ogni utente è un oggetto, le chiavi presenti all'interno dell'oggetto
- `l` che indica se un utente è loggato oppure no;
- `nc` che indica il nome e cognome dell'utente;
- `p` che indica la password dell'utente;
- il vettore `prenotazione` che indica le prenotazioni effettuate dall'utente.

A sua volta, il vettore `prenotazione` è composto da:
- `r` che è il nome del ristorante;
- `g` il giorno;
- `o` l'orario;
- `t` il numero di posti prenotati.

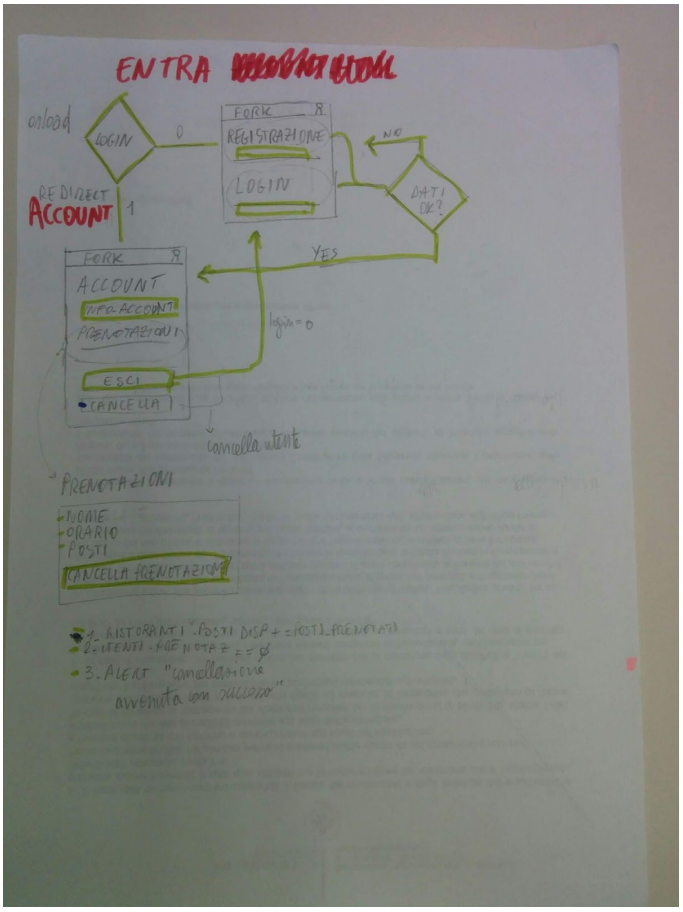
Esempio: `[{"l":1,"nc":"roberto pinotti","p":"ciaonene","prenotazione":[{"r":"Roberto Pizza","g":"0","o":"0","t":"2"}]}, … ]`

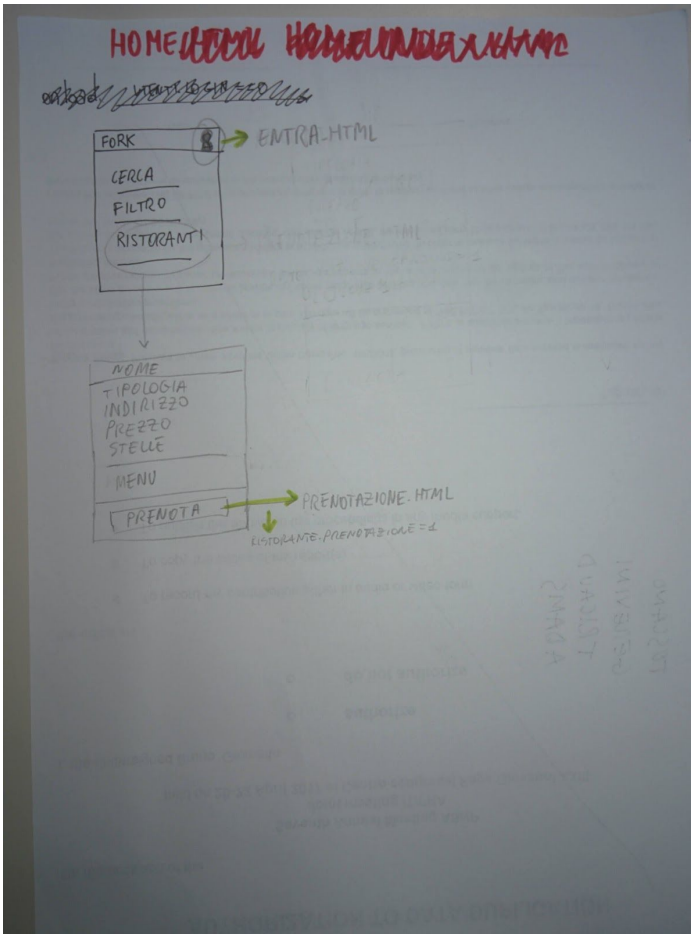
Alcuni schemi iniziali sulla gestione delle due strutture dati:

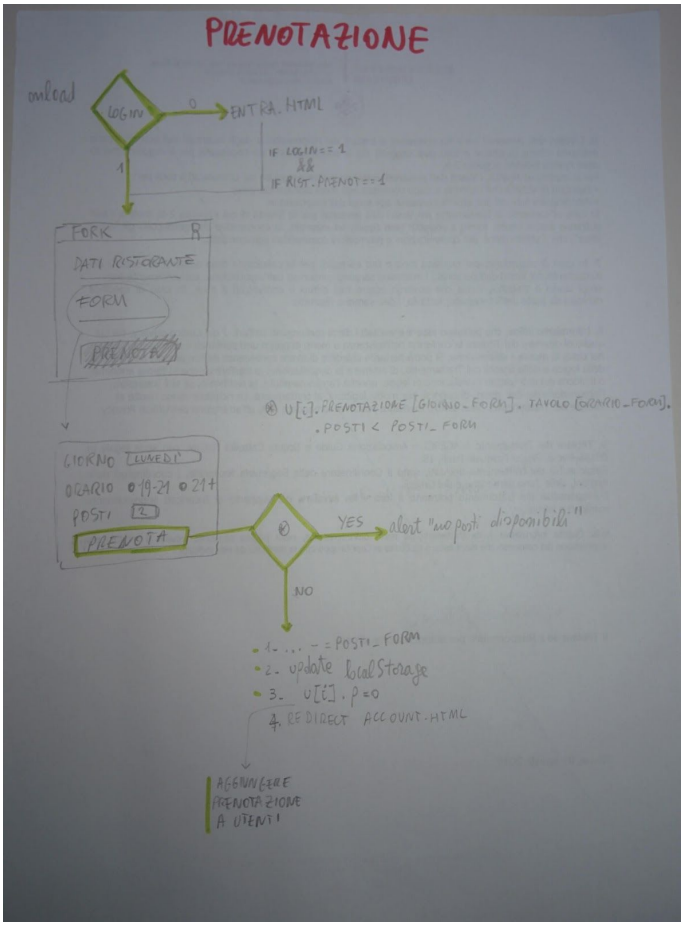


5) Implementazione delle pagine Web

Tutto inzia con degli schizzi:







Tutte le pagine

Ogni pagina web è composta da una navbar e dal resto della pagina che viene generato o staticamente o dinamicamente. Il contenuto della navbar viene generato dinamicamente da una funziona Javascript (*stampaNavbar()*): in questo modo posso far apparire il nome utente, se loggato, affianco all'icona del mezzobusto.

Le funzioni che devono stampare dati, come *stampaNavbar()* o *stampaRistoranti()*, sono inserite nella funzione *onLoad()* presente nel tag body, così da lanciare le funzioni una volta che viene caricata la pagina.

index.html

I dati dei ristoranti vengono stampati dinamicamente grazie alla funzione *stampaRistoranti()* che prende i valori nel localStorage e li stampa dentro delle card (panel in Bootstrap). I dati vengono presi tramite il comando `var vristoranti = JSON.parse(localStorage.getItem("ristoranti"))`.

L'altra funzione presente nella pagina pricipale è quella di ricerca: i valori inseriti dall'utente nel form vengono passati alla funzione *ricercaRistoranti()*. Le variabili `x,y,z,k` servono per controllare che il valore inserito dall'utente combaci con qualche valore nel ristorante. Ad esempio `x = RegExp(nomeris, "ig").test(vristoranti[i].nome)` controlla che il valore inserito dall'utente (`nomeris`) sia contenuto all'interno del nome di un ristorante. Le variabili `nomeris,luogoris,tiporis,prezzoris` mi servono per segnare il fatto che un utente lascia vuoto un campo di ricerca: se il campo è vuoto, imposto il valore a true così da entrare nell'if di stampa: `if ( nomeris && x && luogoris && y && tiporis && z && prezzoris && k)`.

`if (typeof(localStorage.ristoranti) == "undefined") { localStorage.setItem("ristoranti", JSON.stringify(ristoranti));}` serve per inviare al localStorage il contenuto del file JSON, solo se non già inviato precedentemente.

prenotazione.html

La pagina prenotazione è composta dalle informazioni del singolo ristorante e dalla possibilità, se loggati, di prenotare un tavolo.

la funzione *getFromUrl()* serve per recuperare l'id del ristorante passato tramite url dalla home: in questo modo so quale ristorante vuole prenotare l'utente e stampo le informazioni giuste.

stampaDatiRistorante() funziona similmente a *stampaDatiRistorante()* di `index.html`, con l'aggiunta del menu e dei posti disponibili. Sia il menu che i posti disponibili sono contenuti dentro un vettore, quindi la stampa necessita di un doppio ciclo for. All'interno di questa funzione è presente il codice per la stampa della card di prenotazione: se un utente è loggato (`vutenti[i].l==1`) allora viene visualizzato il form di prenotazione, altrimenti viene visualizzato un messaggio che invita a registrarsi o a fare login.

Se un utente è loggato, può utilizzare la funzione *prenotaRistorante()*: la funzione controlla che il ristorante abbia posto nel giorno e nell'orario selezionato e se sì allora decrementa i tavoli e salva la prenotazione nella struttura dati dell'utente, se no viene visualizzato un messaggio di errore.
Per salvare i dati nel localStorage uso il comando `localStorage.ristoranti = JSON.stringify(vristoranti)` per i ristoranti, `localStorage.utenti = JSON.stringify(vutenti)` per gli utenti.

La funzione *vaiAForm()* serve solo per il mobile: per una questione di usabilità faccio visualizzare in cima alla pagina un bottone che invita alla prenotazione e, se premuto, manda al form di prenotazione (altrimenti non visibile perché in fondo alla pagina).

registrazione.html e entra.html

La funzione *checkLogin()* presente sia in `registrazione.html` che in `entra.html` serve per controllare se l'utente è già loggato: se sì, allora avviene un redirect verso la pagina `account.html`.

La funzione *registrazioneUtente()* registra i dati inseriti dall'utente nel localStorage utenti. In particolare `vutenti[vutenti.length] = datiform` inserisce i dati nel nuovo utente nell'ultima posizione del vettore utenti nel localStorage. La funzione *uguali()* controlla che non ci siano altri utenti con lo stesso nome e cognome, chaive primaria del database. Se l'azione è andata a buon fine, l'utente viene indirizzato alla pagina `account.html`.

La funzione *loginUtente()* controlla tramite *uguali_psw()* se esiste nel database un utente con le stesse credenziali inserite dall'utente, se esiste allora l'utente viene indirizzato alla pagina `account.html`.

account.html

La pagina visualizza le prenotazione e i dati dell'utente.

Le prenotazioni vengono visualizzate solo se presenti, tramite la funzione *stampaPrenotazioniUtente()*.

I dati vangono stampati tramite la funzione *stampaDatiUtente()*. L'utente ha la possibilità di modificare sia il nome e cognome sia la password. Per fare ciò ho creato due pagine: `modificanomecognome.html` e `modificapassword.html`. Due semplici form che controllano che le credenziali inserite dall'utente siano corrette e se sì le aggiorna.

Alte funzionalità presenti nella pagina `account.html`:
- Un utente può cancellare il proprio account. La procedura funziona così: carico i dati dal localStorage, cerco l'utente loggato, rimuovo i valori con *splice()* (splice accetta due parametri: il primo indica la posizione da cancellare, il secondo indica quanti dati vanno cancellati), risalvo tutto nel localStorage.
In questo caso ho deciso di NON riassegnare i posti prenotati al ristorante: questo perché, magari, un utente si registra al servizio solo per prenotare un ristorante, ma poi non ha interesse nel tenere l'account attivo.
- Uscire dal proprio account (`vutenti[i].l==0`).
- Cancellare una prenotazione: i posti vengono riassegnati al ristorante. La dinamica è simile a quella della cancellazione dell'account.

La cartella /en/

Ho deciso di creare una copia del sito in lingua inglese. The fork in lingua inglese è accessibile all'indirizzo `/en/index.hmtl` oppure premendo la bandiera della Gran Bretagna nella navbar. Il funzionamento è identito a quanto descritto in precedenza: anche il localStorage e il JSON è lo stesso.

Ho aggiunto nel JSON tre nuovi elementi:
- Il menu in inglese si chiama `menuen`;
- Il tipo di ristorante si chiama `type`;
- Il tipo di sconto si chiama `type`.