



# **lil.deliverer**

by Affuso, Prestia, Sofio

## **lil.deliverer**

lil.deliverer è una web application per la gestione delle consegne a domicilio: disponibili grazie a diversi ristoranti partner. Sviluppata attraverso diverse tecnologie:

- Javascript
- bootstrap
- CSS
- HTML
- Google maps APIs

Places API: <https://developers.google.com/maps/documentation/places/web-service/overview>

Javascript API: <https://developers.google.com/maps/documentation/javascript/overview>

Distance matrix API: <https://developers.google.com/maps/documentation/distance-matrix/overview>

Geocoding API: <https://developers.google.com/maps/documentation/geocoding/overview>

- TailWind

TailWind pur non essendo tra le tecnologie prescritte per lo sviluppo è stato utilizzato perché si tratta di un framework davvero molto comodo ed efficace per la semplificazione della scrittura di classi bootstrap. L'unica sua finalità è rivolta all'interfaccia grafica e non interferisce in alcun modo con la logica delle funzionalità.

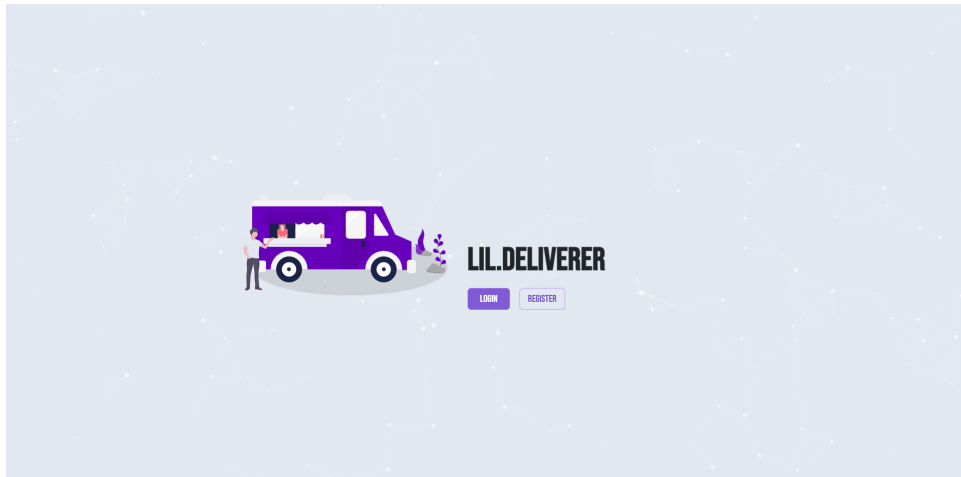
È stato molto proficuo l'utilizzo di GitHub, che ha concesso agli sviluppatori di accelerare e ottimizzare la realizzazione del progetto.

Le funzionalità richieste erano molteplici e di diversa complessità e nel seguente documento saranno elencate le scelte implementative effettuate per lo sviluppo delle varie richieste.

## **Struttura**

L'intero progetto è stato creato e basato su un'interfaccia minimale e molto intuitiva, per regalare un'esperienza quanto più semplificata a qualsiasi tipo di utente, senza, naturalmente, tralasciare la qualità.

## **Landing page**

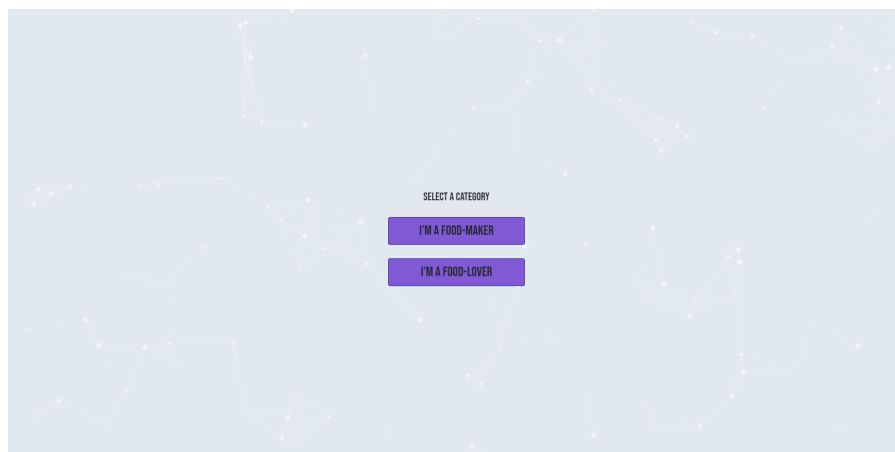


Pagina index.html

L'immagine precedente illustra la pagina di approdo, dove l'utente può effettuare il login oppure registrarsi alla nostra app.

### Category selection page

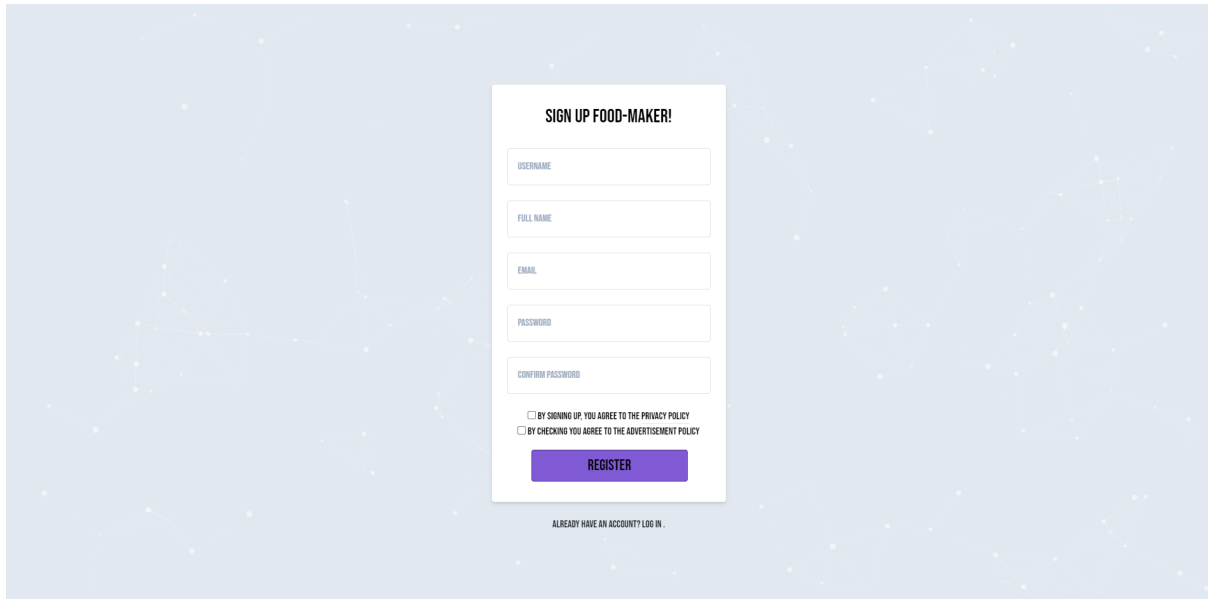
Pagina in comune presente sia per il login che per la registrazione



Pagina selezione categoria

L'immagine presente illustra la sezione dove l'utente può selezionare la categoria (rispettivamente ristoratore e consumatore) secondo la quale vuole effettuare il login o la registrazione, infatti al variare della classe di appartenenza le informazioni richieste per l'iscrizione saranno diverse. Tuttavia per il login, in entrambi i casi, saranno richiesti solamente username e password.

### Registration page - Food maker



**SIGN UP FOOD-MAKER!**

USERNAME

FULL NAME

EMAIL

PASSWORD

CONFIRM PASSWORD

☐ BY SIGNING UP, YOU AGREE TO THE PRIVACY POLICY  
☐ BY CHECKING YOU AGREE TO THE ADVERTISEMENT POLICY

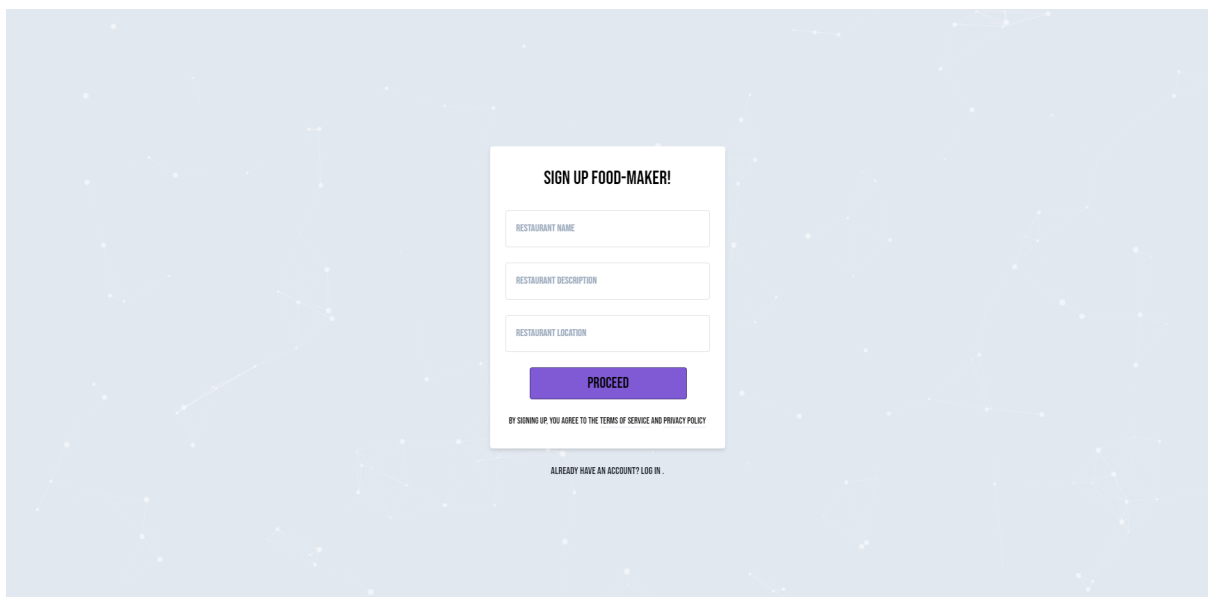
**REGISTER**

ALREADY HAVE AN ACCOUNT? LOG IN

Registration page 1 - Food maker

In questa sezione si può vedere la prima parte della registrazione di un ristorante, dove i campi richiesti sono username (necessariamente univoco), il nome completo, l'email, la password (minimo otto caratteri), importanti sono le check box dove si accettano o meno i consensi per il trattamento della privacy e dei dati a fini promozionali.

Premendo il pulsante "register", se tutti i campi sono stati compilati correttamente, si potrà accedere alla seconda parte della registrazione.



**SIGN UP FOOD-MAKER!**

RESTAURANT NAME

RESTAURANT DESCRIPTION

RESTAURANT LOCATION

**PROCEED**

BY SIGNING UP, YOU AGREE TO THE TERMS OF SERVICE AND PRIVACY POLICY

ALREADY HAVE AN ACCOUNT? LOG IN

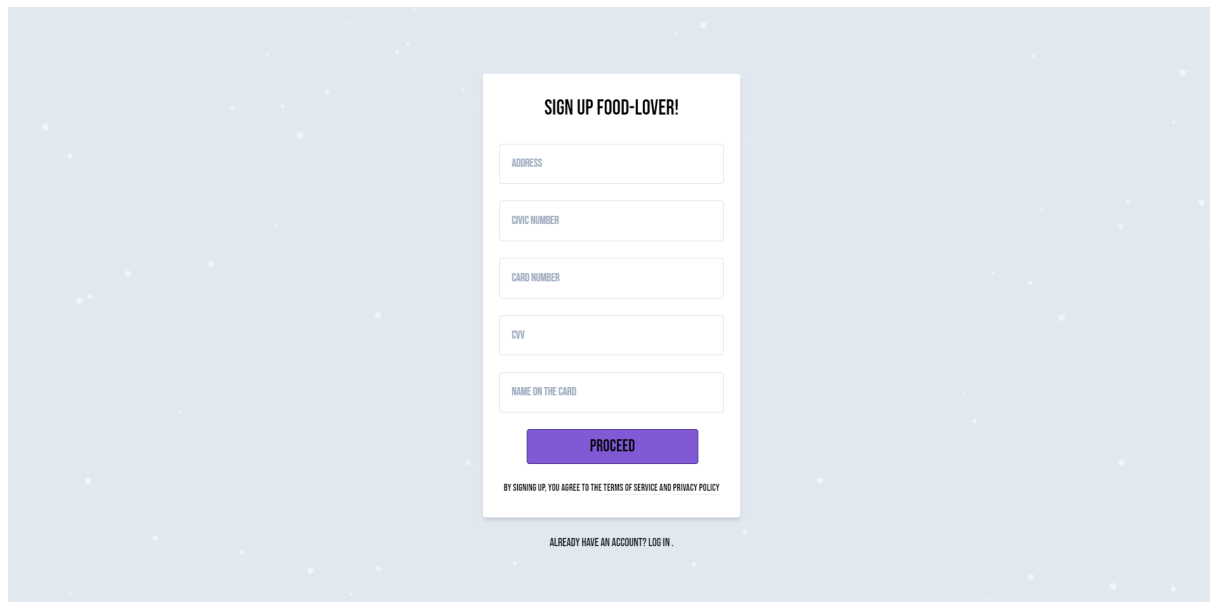
Registration page 2 - Food maker

A questo punto ci verrà chiesto di inserire il nome del ristorante, una breve descrizione del ristorante e in fine l'indirizzo del locale, che viene compilato quasi automaticamente grazie all'API di Google maps.

Una volta effettuata la registrazione il ristorante sarà visibile e soprattutto usufruibile da tutti i consumatori. Per ora il ristorante deve limitarsi a registrare il suo locale, ma in futuro verrà implementato tutto ciò che riguarda la gestione dei Food makers.

### Registration page - Food lover

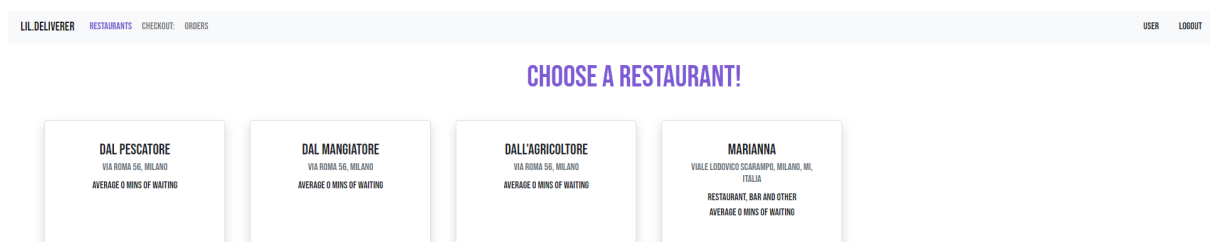
La registrazione è la medesima tranne che per i seguenti campi presenti nella seconda pagina.

A registration form titled "SIGN UP FOOD-LOVER!" is centered on a light blue background with white stars. The form contains five input fields: "ADDRESS", "CIVIC NUMBER", "CARD NUMBER", "CVV", and "NAME ON THE CARD". Below these fields is a purple "PROCEED" button. At the bottom of the form, it says "BY SIGNING UP, YOU AGREE TO THE TERMS OF SERVICE AND PRIVACY POLICY". Below the form, there is a link that says "ALREADY HAVE AN ACCOUNT? LOG IN."

Registration page 2 - Food lover

Anche qui il campo dell'indirizzo viene compilato quasi automaticamente grazie all'API di Google maps.  
Card number e CVV devono rispettare il formato, rispettivamente, di 16-13 caratteri e 3 caratteri.

## Main page



Main page - choose a restaurant

La seguente schermata mostra la main page dove l'utente può scegliere un ristorante dal quale ordinare uno o più piatti:

## CHOOSE A RESTAURANT!

<b>DAL PESCATORE</b> VIA ROMA 56, MILANO AVERAGE 40 MINS OF WAITING	<b>DAL MANGIATORE</b> VIA ROMA 56, MILANO AVERAGE 0 MINS OF WAITING	<b>DALL'AGRICOLTORE</b> VIA ROMA 56, MILANO AVERAGE 30 MINS OF WAITING
---	---	--

## DAL PESCATORE

SPAGHETTI ALLE VONGOLE	VEG	BIO	15 MINS	5.76€	ADD
STINCO DI MAIALE			5 MINS	7.65€	ADD
MACEDONIA	VEG		5 MINS	2.64€	ADD

## Plates selection

Si nota immediatamente la possibilità di aggiungere un piatto al carrello, premendo il tasto add. Inoltre è presente un'etichetta che informa il consumatore se il prodotto scelto è biologico e/o vegano.

Come da richiesta l'utente può ordinare piatti solo da un solo ristorante per ogni ordine.

## Checkout page

## YOUR ORDER!

SPAGHETTI ALLE VONGOLE	15 MINS	5.76€	ADD
STINCO DI MAIALE	5 MINS	7.65€	ADD
<input type="radio"/> WITHDRAWAL ON THE SPOT <input type="radio"/> HOME DELIVERY	20 MINS	13.41€	

CHECKOUT

## Checkout page - with Plates

In checkout il cliente può eliminare un piatto, visualizzare il prezzo totale e il tempo necessario per la preparazione, nella sezione successiva potrà anche verificare la distanza tra il suo indirizzo e il locale e il sovrapprezzo scaturito dai km.

Per concludere l'ordine basta selezionare la modalità:

- consegna a domicilio (solo per distanze inferiori a 50 km)
- ritiro in loco

e in fine premere su checkout.

YOUR CHECKOUT IS EMPTY! 😞

Checkout page - without Plates

## Orders page

### YOUR ORDERS!

RECIPIENT: LUKA

SENDER: DALL'AGRICOLTORE

ADDRESS: VIA ROMA, TORINO, TO, ITALIA, 12

ADDRESS: VIA ROMA 56, MILANO

POLENTA PASTICCIATA

15 MINS

5.76€

POLENTA PASTICCIATA

15 MINS

5.76€

WITHDRAWAL ON THE SPOT

140 KM

1 ORA 39 MIN

30 MINS


11.52€ + 0€ FEE

Orders page

Nella soprastante immagine è presente la pagina di riepilogo degli ordini avvenuti, completa di informazioni quali: l'indirizzo di entrambe le parti, i piatti ordinati, i prezzi, il totale della spesa, compreso di sovrapprezzo dovuto alla distanza, la distanza stessa e tempo totale della coda più tempo di preparazione.

## User info page

## USER INFORMATION

 LUCA ROSSI FOOD LOVER	INFORMATION	
	EMAIL LUCAROSSIGMAIL.COM	USERNAME LUKA
	PAYMENT INFORMATION	
	NAME ON THE CARD ROSSI LUCA	CARD NUMBER *****543
	POSITION	
	TOWN TORINO, TO, ITALIA	ADDRESS VIA ROMA, 12
	<a href="#">MODIFY</a>	<a href="#">DELETE ACCOUNT</a>

user info page

In questa sezione l'utente può visualizzare le sue informazioni e attraverso i pulsanti "modify" e "delete account", ha la possibilità, rispettivamente, di modificare i propri dati o eliminare l'account.

## Modification page

La sezione modifica è stata basata, naturalmente, sulla grafica e su alcune funzionalità della registrazione.

MODIFICATION FOOD-LOVER!

☐ YOU AGREE TO THE PRIVACY POLICY

☐ BY CHECKING YOU AGREE TO THE ADVERTISEMENT POLICY

[MODIFY](#)

Modification page 1

Nella seguente pagina, così come per la successiva, i form vengono automaticamente compilati prelevando i dati dal session storage. L'utente può modificare qualsiasi campo stando ovviamente attento ai formati corretti di password, mail, username, numero carta e carta.

**MODIFICATION FOOD-LOVER!**

VIA ROMA, TORINO, TO, ITALIA

VIA ROMA TORINO, TO, ITALIA powered by Google

1234567854567543

123

ROSSI LUCA

PROCEED

BY MODIFYING, YOU AGREE TO THE TERMS OF SERVICE AND PRIVACY POLICY

ALREADY HAVE AN ACCOUNT? LOG IN.

Modification page 2

In qualsiasi zona del sito si può sempre effettuare il log out premendo in alto a destra l'apposito pulsante.

## Funzionalità - Elementi salienti

### Registrator page

```
function addressMaps(){

    //codice per l'auto completamento del form riguardante l'indirizzo usando l'API

    let autocomplete;
    let address;

    function initAutocomplete() {
        //Funzione che scrive all'interno del form address
        address = document.querySelector("#address");

        //Funzione di Google per autocompilazione
        autocomplete = new google.maps.places.Autocomplete(address, {
            fields: ["address_components", "geometry"],
            types: ["address"],
        });
        address.focus();

        autocomplete.addListener("place_changed", fillInAddress);
    }

    function fillInAddress() {

        const place = autocomplete.getPlace();
        let address1 = "";
        let postcode = "";

        for (const component of place.address_components) {
            const componentType = component.types[0];

            /*Attraverso questo switch vado a selezionare i valeri elementi
            necessari per compiere il form*/

            switch (componentType) {
                case "street_number": {
                    address1 = `${component.long_name} ${address1}`;
                    break;
                }

                case "route": {
                    address1 += component.short_name;
                    break;
                }
            }
        }
    }
}
```



```

    case "postal_code": {
      postcode = `${component.long_name}${postcode}`;
      break;
    }

    case "postal_code_suffix": {
      postcode = `${postcode}-${component.long_name}`;
      break;
    }
    case "locality":
      address1 += component.long_name;
      break;

    case "administrative_area_level_1": {
      address1 += component.short_name;
      break;
    }
    case "country":
      address1 += component.long_name;
      break;
  }
}

}

initAutocomplete(); /*una volta ottenuti tutti gli elementi
necessari vado a chiamare la funzione
che scriverà all'interno del form address*/
}

```

Questo codice è la funzione, basata sull'API di Google maps, costruita per l'auto completamento dei form. Semplicemente prende in considerazione diverse informazioni prelevate dai suggerimenti di maps come: la nazione, la via, il paese, la località.

```

//eventi e funzioni scatenate in seguito alla pressione del pulsante di registrazione
lover.addEventListener('click', function() {
  console.log('lover :')
  main.innerHTML = formL
  var registrationLover = document.getElementById('registrationlover');
  registrationLover.addEventListener('click', event => {
    event.preventDefault()
    user = validatorProfile()
    const error = document.getElementById('error')
    error.innerHTML = ''
    //se user non ha riscontrato errori durante la validazione sostituisce main con la pagina successiva per la registrazione
    if (user != 'check your password, it must be the same of confirm password and at least 8 char long' && user != 'insert valid u
      main.innerHTML = formLInfos

    addressMaps();

    registrationLover = document.getElementById('registrationlover2')
    //eventi e funzioni scatenate in seguito al click del pulsante finale per la registrazione, funzioni asincrone
    registrationLover.addEventListener('click', async event => {
      //previene il refresh della pagina
      event.preventDefault()
      //sono funzioni fondamentali per la corretta esecuzione del codice dunque sono in stato di await e il resto del progra
      await validatorLover(user)
      if(user.address.position){
        await storeUser(user)
        await storeLog(user)
        window.location.replace('/views/main/main.html')
      }
    })
  } else error.innerHTML = user
})
});

```

Di seguito verranno elencate le funzioni di validazione e registrazione dell'utente.

```

//Funzione per la validazione delle informazioni necessarie alla registrazione
function validatorProfile() {
  let username = document.getElementById('username').value
  let fullname = document.getElementById('fullname').value
  let email = document.getElementById('email').value
  let password = document.getElementById('password').value
  let confirmed_password = document.getElementById('confirm_password').value
  let usersArr = JSON.parse(localStorage.getItem('users'))
  let same = false

  if(usersArr)

```

```

usersArr.map(utente => {if(utente.username == String(username)) {same = true} })
//Vari controlli per i campi inseriti dall'utente
if(!same){
if (username.length > 3 && fullname.length > 3 && email.includes('@')) {
  if (password.length >= 8 && password == confirmed_password) {
    if (!check) {
      return user = {
        username: username,
        fullname: fullname,
        email: email,
        password: password,
        maker: check,
        payment: {},
        address: {coordinate: {}},
        privacy: {
          offerte_personalizzate: document.getElementById('advertisement').checked,
          consenso_privacy: document.getElementById('privacy').checked
        }
      }
    }
  }
  if (check) {
    return user = {
      username: username,
      fullname: fullname,
      email: email,
      password: password,
      maker: check,
      restaurant: {piatti_ordinabili: [{}]},
    }
  }
} else { return 'check your password, it must be the same of confirm password and at least 8 char long' }
} else { return 'insert valid username and email' }
}else {return 'This username already exists'}
}

```

Questa funzione è particolarmente importante perché controlla anche che il nome utente sia univoco. Se le condizioni non sono soddisfatte verrà ritornato un messaggio d'errore.

```

//Altri controlli per la validazione del consumatore
async function validatorUser(user) {
  const error = document.getElementById('errorL2')
  const cardNumber = document.getElementById('cardNumber').value
  const cardCvv = document.getElementById('cardCvv').value
  const cardName = document.getElementById('cardName').value
  const position = document.getElementById('address').value
  const civic = document.getElementById('civic').value

  if (position && cardNumber && cardCvv && cardName && civic) {
    if (Number(cardNumber.length) <= 16 && Number(cardNumber.length) >= 13 && Number(cardCvv.length) == 3) {
      user.payment.cardNumber = cardNumber
      user.payment.cardCvv = cardCvv
      user.payment.cardName = cardName
      user.address.position = position
      user.address.civic = civic
    } else { error.innerHTML = 'Please fill every single field correctly' }
  } else { error.innerHTML = 'Please fill every single field' }
}

```

```

//Mette l'utente appena registrato, con le sue informazioni, nel session storage
async function storeLog(user){
  sessionStorage.setItem('logged', JSON.stringify(user))
}

//Mette l'utente appena registrato, con le sue informazioni, nel local storage
function storeUser(user){
  let usersArr
  if(localStorage.getItem('users') === null){
    usersArr = [];
  }else{
    usersArr = JSON.parse(localStorage.getItem('users')) //converte da json a js object
  }

  console.log(user)
  console.log(usersArr)

  usersArr.push(user)
}

```

```

    localStorage.setItem('users', JSON.stringify(usersArr)) //converte da js object a json
  }

```

## Login page

```

//funzione login
function loginHandler(){
  //prendo i valori dei form username e password
  let username = document.getElementById('username').value
  let password = document.getElementById('password').value
  let checkLog = false

  //metti in un array tutti gli utenti presenti nel local storage
  let userArr = JSON.parse(localStorage.getItem("users"))
  //se userArr esiste lo scorro e cerco le credenziali giuste per fare il login
  if(userArr){
    userArr.forEach(user => {
      if(user.username == String(username) && user.password == String(password) && user.maker == checkMaker){
        userLog = user
        checkLog = true
      }
    })
    return checkLog
  }
}

```

## Loader page

```

//creo un array dentro al quale inserisco gli elementi presi dal file JSON
const usersArr = []
//Fetch ci permette di leggere i dati dal JSON, usiamo una promise e dei then
await fetch('data/users.json').then(response => response.json()).then(data => {
  data.users.forEach(user => {user.maker = false; usersArr.push(user)}))

await fetch('data/makers.json').then(response => response.json()).then(data => {
  data.makers.forEach(maker => {maker.maker = true; usersArr.push(maker)}))

//Eliminiamo gli utenti già presenti
registeredUsers.forEach(user => {usersArr.forEach((userF, index) => {if(user.username == userF.username) usersArr.splice(index, 1)})})

if(usersArr.length > 0) usersArr.forEach(user => {registeredUsers.push(user)})
//Mettiamo nel local storage gli utenti presenti nel json
localStorage.setItem('users', JSON.stringify(registeredUsers))

```

Il file Loader ci consente di prelevare le informazioni dal JSON e metterle nel local storage

## Modification page

```

//cerchiamo nell'array di utenti il nome utente della persona da modificare e lo eliminiamo, sostituendolo con le modifiche effettuate
usersArr.forEach((user, index) => {
  {
    console.log(user.username)
    if(user.username == oldUsername){
      usersArr.splice(index, 1)
    }
  })
  usersArr.push(user)
  localStorage.setItem('users', JSON.stringify(usersArr))

```

Del Modification page è presente solamente questo metodo perché è molto simile al codice per la registrazione.

## Orders page

```

let ordersContainer = document.getElementById('orders')
let checkoutN = document.getElementById('checkoutN')

document.addEventListener('DOMContentLoaded', event => {
  checkoutN.innerHTML = sessionStorage.getItem('checkoutN')
})

```

```

displayOrders()

function displayOrders(){
  let totalOrders = JSON.parse(localStorage.getItem('ordersInQueue'))
  let loggedUser = JSON.parse(sessionStorage.getItem('logged'))
  let userOrders = []

  if(totalOrders == null){
    document.getElementById('title').textContent = 'You have no orders! 🍷'
  }

  totalOrders.forEach(order => {
    if(order.recipient === loggedUser.username){
      userOrders.push(order)
    }
  })

  userOrders.forEach(order => {
    console.log(order)

    ordersContainer.innerHTML += `
    <div class="flex mt-32 mx-10">
      <p class="flex-1 text-xl">Recipient: ${order.recipient}</p>
      <p class="w-1/10 text-xl">Sender: ${order.sender}</p>
    </div>
    <div class="flex mt-10 mx-10">
      <p class="flex-1 text-xl">address:${order.recipientAddress}</p>
      <p class="w-1/10 text-xl">address:${order.senderAddress}</p>
    </div>`

    order.orders.forEach(plate => {
      ordersContainer.innerHTML += `
      <div class="flex mt-10 mx-10">
        <p class="flex-1 text-2xl">${plate.name}</p>
        <p class="w-1/6 text-l text-gray-600">${plate.minutes}</p>
        <p class="w-1/10 mr-5 text-2xl">${plate.price}</p>
      </div>`
    })
  })
}

```

Questa pagina ci consente di mostrare gli ordini effettuati dall'utente

## Main page

Main page si compone di due funzioni fondamentali, displayResturant e displayPlates.

```

function displayRestaurants(){
  let data = JSON.parse(localStorage.getItem('users'))
  let ordersInQueue = JSON.parse(localStorage.getItem('ordersInQueue'))
  let timeQueue = 0

  let makers = []
  data.forEach(maker => {
    if(maker.maker) makers.push(maker)
  })

  makers.map(maker => {

    if(ordersInQueue){
      ordersInQueue.forEach(order => {
        if(order.sender == maker.restaurant.name && order.senderAddress == maker.restaurant.address){
          timeQueue += 10
        }
      })
    }

    if(maker.restaurant.description == undefined) maker.restaurant.description = ""
    card = `<div>
      <div class="card rest p-3 shadow" style="width: 18rem; margin-right: 20px; margin-left: 20px; max-height: 200px; height:
        <div class="card-body">
          <h5 class="card-title text-2xl" id='name'>${maker.restaurant.name}</h5>
          <h6 class="card-subtitle mb-2 text-muted">${maker.restaurant.address}</h6>
          <p class="card-text">${maker.restaurant.description}</p>
          <p class="card-text" id="user">${maker.username}</p>
          <p class="card-text">average ${timeQueue} mins of waiting</p>
        </div>
      </div>`
    restaurantContainer.innerHTML += card
    timeQueue = 0
  })
}

```

```
function displayPlates(plates){
  let veg = ''
  let bio = ''
  plates.forEach(plate => {
    if(plate.nome_piatto != undefined){

      if(plate.vegetariano) veg = 'veg'
      if(plate.biologico) bio = 'bio'
      //sostituisce il contenuto con ciò che gli passiamo
      platesContainer.innerHTML += `
<div class="flex mt-10 mx-10">
  <p class="flex-1 text-2xl">${plate.nome_piatto}</p>
  <p class="w-1/6 text-l text-green-600">${veg}</p>
  <p class="w-1/6 text-l text-green-600">${bio}</p>
  <p class="w-1/6 text-l text-gray-600">${plate.tempo_preparazione} <span>mins</span></p>
  <p class="w-1/10 mr-5 text-2xl">${plate.prezzo}€</p>
  <button class="bg-purple-600 hover:bg-purple-800 text-white font-bold px-2 py-1 rounded mb-2 addBtn">Add</button>
</div>
<hr class="mx-10">`
      veg = ''
      bio = ''
    }else{
      platesContainer.innerHTML += `
<div class="flex mt-10 mx-10">
  <p class="flex-1 text-2xl">No plates yet</p>
</div>
<hr class="mx-10">`
    }
  })
}
```

Il codice di questa pagina si compone di elementi e funzioni già utilizzate e spiegate nelle altre pagine. Semplicemente usiamo questo codice per mostrare i ristoranti e i relativi piatti ordinabili nella pagina principale.

## Log out page

```
let logout = document.getElementById("logout")

logout.addEventListener("click", () => {
  sessionStorage.removeItem("logged");
})
```

Attraverso questa funzione eliminiamo l'utente loggato dal session storage in modo tale da rimuovere l'accesso durante la sessione corrente.

## Info user page

```
//preleviamo le info dell'utente loggato dal session storage e lo mettiamo in un array
let loggedUser = JSON.parse(sessionStorage.getItem("logged"))

document.addEventListener('DOMContentLoaded', event => {
  checkoutN.innerHTML = sessionStorage.getItem('checkoutN')
  if(checkoutN.innerHTML == 0){
    document.getElementById('formCont').innerHTML = ``
  }
})

document.getElementById("name").innerHTML = String(loggedUser.fullname)
document.getElementById("email").innerHTML = String(loggedUser.email)
document.getElementById("username").innerHTML = String(loggedUser.username)
document.getElementById("cardname").innerHTML = String(loggedUser.payment.cardName)

//splitto in un array di char le cifre che appartengono al card number
let cardNumber = String(loggedUser.payment.cardNumber)
let splittedCardNumber = cardNumber.split('')

//in base alla lunghezza dell'array obliero le cifre ad eccezione delle ultime tre
if(cardNumber.length == 13){
  cardNumber = "*****" + splittedCardNumber[10] + splittedCardNumber[11] + splittedCardNumber[12]
}
else{
  cardNumber = "*****" + splittedCardNumber[13] + splittedCardNumber[14] + splittedCardNumber[15]
}
```

```

document.getElementById("cardnumber").innerHTML = cardNumber

//divido l'indirizzo in modo tale da mostrarlo in modo più chiaro ed estetico
let position = String(loggedUser.address.position)
let positionSplitted = position.split(',')

document.getElementById("town").innerHTML = positionSplitted[1] + ", " + positionSplitted[2] + ", " + positionSplitted[3]

positionSplitted.splice(1, 3)

let arrayUnito = positionSplitted.join(',')

console.log(arrayUnito)

document.getElementById("address").innerHTML = arrayUnito + ", " + loggedUser.address.civic

```

Il precedente codice consente di visualizzare le informazioni dell'utente nell'apposito box.

## Delete function

```

let del = document.getElementById("deleteAccount")

del.addEventListener("click", () => {
  var conformation = confirm("do you really want to delete your account?")
  //scorro l'array contenente gli utenti del local storage e cerco l'utente confrontandolo con quello presente nel session
  if(conformation){
    usersArr = JSON.parse(localStorage.getItem("users"))
    currentUsers = JSON.parse(sessionStorage.getItem("logged"))
    usersArr.forEach((user, index) => {if(user.username == currentUsers.username) usersArr.splice(index, 1)})
    localStorage.setItem("users", JSON.stringify(usersArr))
    sessionStorage.removeItem("logged")
    location.replace("../index.html")
  }
})

```

Funzione che consente all'utente di eliminare il proprio account, cancellandolo sia dal local storage che dal session storage.

## Checkout page

```

async function distanceCalclater(recipientAddress, senderAddress){

  r = recipientAddress.split(',')
  r.splice(r.length, 1)

  recipientAddress = r.join(',')

  s = senderAddress.split(',')
  s.splice(r.length, 1)

  senderAddress = s.join(',')

  console.log(recipientAddress, senderAddress)
  //Pongo una richiesta all'API di geocode di maps per ottenere le coordinate del ristorante e del cliente
  var rGeo = await fetch('https://maps.googleapis.com/maps/api/geocode/json?address=${recipientAddress.trim()}&key=AIzaSyBIO5Nb
  var sGeo = await fetch('https://maps.googleapis.com/maps/api/geocode/json?address=${senderAddress.trim()}&key=AIzaSyBIO5Nb4gE

  let rLat = rGeo.results[0].geometry.location.lat
  let rLon = rGeo.results[0].geometry.location.lng

  let sLat = sGeo.results[0].geometry.location.lat
  let sLon = sGeo.results[0].geometry.location.lng

  //uso la funzione di google per calcolare la distanza e il tempo di percorrenza tra i due luoghi
  const matrix = new google.maps.DistanceMatrixService();
  //inizializzo una promise
  var d = $.Deferred();
  //calcolo effettivamente la distanza
  matrix.getDistanceMatrix({
    origins: [new google.maps.LatLng(rLat, rLon)],
    destinations: [new google.maps.LatLng(sLat, sLon)],
    travelMode: google.maps.TravelMode.DRIVING,
  }, function(response, status) {
    if (status != google.maps.DistanceMatrixStatus.OK) { //OK == HTTP status 200 != OK means 400/500/300
      d.reject(status);
    }
  });
}

```

```

    } else {
      //Se il server, o meglio l'http ha responso positivo (200) salvo la risposta
      d.resolve(response);
    }
  });
  //ritorno la promise
  return d.promise()
}

```

```

function displayOrders(){
  let ordersArr = JSON.parse(sessionStorage.getItem('checkout'))
  ordersArr.forEach(ordine => {
    orders.innerHTML += `
    <div class="flex mt-10 mx-10">
      <p class="flex-1 text-2xl">${ordine.name}</p>
      <p class="w-1/6 text-l text-gray-600">${ordine.minutes}</p>
      <p class="w-1/10 mr-5 text-2xl">${ordine.price}</p>
      <button class="delete bg-purple-600 hover:bg-purple-800 text-white font-bold px-2 py-1 rounded mb-2">✕</button>
    </div>
    <hr class="mx-10">`

    order.orders.push(ordine)

    totalPrice.innerHTML = (Number(totalPrice.innerText) + Number(ordine.price.replace('€', '').trim())).toFixed(2)
    totalTime.innerHTML = Number(totalTime.innerText) + Number(ordine.minutes.replace(' mins', ''))
  });

  totalPrice.innerHTML += '€'
  totalTime.innerHTML += ' mins'

  deleteButtons = document.getElementsByClassName("delete")
  console.log(deleteButtons)
  Array.from(deleteButtons).forEach(button => {
    button.addEventListener("click", event => {
      let name = event.target.parentNode.children[0].textContent
      let plates = JSON.parse(sessionStorage.getItem("checkout"))
      let newPlates = []
      let check = false
      plates.forEach(plate => {
        if(name == plate.name && !check){
          check = true
        }else{
          newPlates.push(plate)
        }
      })
      sessionStorage.setItem("checkout", JSON.stringify(newPlates))
      let n = sessionStorage.getItem("checkoutN")
      n--
      sessionStorage.setItem("checkoutN", n)
      if(n == 0){
        sessionStorage.removeItem("checkout")
        sessionStorage.removeItem("checkoutN")
        sessionStorage.removeItem("restaurantName")
      }
      window.location.reload()
    })
  })
}

```

Il codice sovrastate consente la visualizzazione degli ordini prima di effettuare il checkout di conferma, nel quale possiamo eliminare un prodotto, scegliere il metodo di consegna/ritiro.

## Authentication function

```

const user = sessionStorage.getItem('logged')

if(!user){
  window.location.replace('../index.html')
}

```

L'antecedente funzione limita la possibilità agli utenti non loggati di muoversi per le pagine della web application.