

Specifiche del progetto di Applicazioni Web e Cloud

@kellygreta

2021-2022

Indice

1	Il progetto in sintesi	2
2	Registrazione	3
3	Accesso	20
4	Creare una playlist	23
5	Cercare tracce	24
6	Cercare album e artisti	29
7	Cercare playlist	32
8	Tracce suggerite	35
9	Profilo	37
9.1	Modifica playlist	39
9.2	Modifica profilo	41

1 Il progetto in sintesi

Il progetto consiste nella realizzazione di una applicazione web Gestione di Comunità Musicali Online (GCMO) che implementa un sito di gestione di playlist musicali sfruttando le API REST del portale Spotify.

L'applicazione prevede una fase di registrazione utente dove vengono collezionati informazioni quali nome utente, indirizzo email, password, e preferenze musicali (generi, canzone e artista preferito).

Dopo aver eseguito l'accesso al sito web è possibile:

- creare playlist musicali inserendo nome, descrizione e tag. Inoltre c'è la possibilità di rendere la playlist pubblica o privata.
- cercare tracce in base al loro titolo
- cercare album e le tracce contenute al suo interno
- cercare artisti e visualizzare le loro tracce più popolari
- cercare e importare le playlist pubbliche di altri utenti
- visualizzare le tracce suggerite in base alle preferenze musicali dell'utente
- modificare ed eliminare le informazioni relative a playlist e utente

2 Registration

Registrazione a GCMO

Nome utente

Email address

Password

Conferma la password

Scegli un artista

add	nome	genere

Scegli i tuoi generi musicali preferiti (max 3)

acoustic	afrobeat	alt-rock	alternative	ambient	anime
black-metal	bluegrass	blues	bossanova	brazil	breakbeat
british	cantopop	chicago-house	children	chill	classical
club	comedy	country	dance	dancehall	death-metal
deep-house	detroit-techno	disco	disney	drum-and-bass	dub
dubstep	edm	electro	electronic	emo	folk
forró	french	funk	garage	german	gospel
goth	grindcore	groove	grunge	guitar	happy
hard-rock	hardcore	hardstyle	heavy-metal	hip-hop	holidays
honky-tonk	house	idm	indian	indie	indie-pop
industrial	iranian	j-dance	j-idol	j-pop	j-rock
jazz	k-pop	kids	latin	latino	malay
mandopop	metal	metal-misc	metalcore	minimal-techno	movies
mpb	new-age	new-release	opera	pagode	party
philippines-opm	piano	pop	pop-film	post-dubstep	power-pop
progressive-house	psych-rock	punk	punk-rock	r-n-b	rainy-day
reggae	reggaeton	road-trip	rock	rock-n-roll	rockabilly
romance	sad	salsa	samba	sertanejo	show-tunes
singer-songwriter	ska	sleep	songwriter	soul	soundtracks
spanish	study	summer	swedish	synth-pop	tango
techno	trance	trip-hop	turkish	work-out	world-music

Scegli una canzone

add	titolo	artista	durata	data di pubblicazione	cover

[Registrati](#) oppure [Accedi](#)

Figure 1: schermata di registrazione

Per accedere a GCMO un utente deve registrarsi attraverso il form della figura-1. Se uno o più campi vengono compilati in modo scorretto, e viene in seguito cliccato il tasto registrati, un avvertimento notificherà l'utente che non può registrarsi. Nella schermata l'utente potrà osservare degli alert più specifici su quali campi deve modificare.

Se invece i campi vengono compilati correttamente viene visualizzato un alert di successo.

```
<button type="submit" class="btn btn-primary" onclick="check();registrati()">Registrati</button>
oppure <a href="homepage.html">Accedi</a>
..
```

Figure 2: funzioni innescate dal clic del tasto registrati

```

function registrati() {
    var users = window.localStorage.getItem('users')

    if (users == null) {
        users = []
    } else {
        users = JSON.parse(users)
    }

    var genres = ""
    var checkedChecks = document.querySelectorAll(".btn-check:checked")
    for (var i = 0; i < checkedChecks.length; i++) {
        if(i==0){
            genres += checkedChecks[i].id
        }else{
            genres += "%2C" + checkedChecks[i].id
        }
    }

    user = {
        nickname: document.getElementById('nome').value,
        email: document.getElementById('email').value,
        password: document.getElementById('password').value,
        favArtist: document.getElementById('artistid').value,
        favTrack: document.getElementById('trackid').value,
        favGenres: genres
    }

    if (findUser(users, user)) {
        alert("Utente già registrato")
    } else if (!minLen("email", 6) || !checkMail("email") || !minLen("nome", 3)
    || !testPass() || !checkPassword() || !minLen("artistid", 1) || !minLen("trackid", 1) || !checkGeneri(".btn-check:checked")) {
        alert("Non è possibile registrare l'utente perché uno dei campi è stato compilato in modo scorretto")
    } else {
        users.push(user)
    }

    window.localStorage.setItem('users', JSON.stringify(users))
}

```

Figure 3: registrati() prende dal local storage l'array degli utenti, se non esiste lo crea. Crea anche un utente che ha come dati quelli inseriti nei vari campi di input della schermata di registrazione: nome, mail, password e artista, generi e canzone preferiti. In seguito controlla che la mail inserita non sia già associata ad un utente. Se uno dei campi non è compilato in modo corretto viene visualizzato un alert. Se invece i campi vengono compilati correttamente l'utente viene aggiunto all'array degli utenti nel local storage.



Figure 4: alert nella schermata di registrazione

```

function check() {
    clearBadge()

    if (!minLen("email", 6)) {
        addBadge("Email troppo corta!");
    }
    if (!checkMail("email")) {
        addBadge("Email non valida!");
    }
    if (!minLen("nome", 3)) {
        addBadge("Nome troppo corto!");
    }
    if (!checkPassword()) {
        addBadge("La password non soddisfa i requisiti dei criteri di password");
    }
    if (!testPass()) {
        addBadge("La password inserita non coincide con la prima!");
    }
    if (!minLen("artistid", 1)) {
        addBadge("Aggiungi il tuo artista preferito!");
    }
    if(!checkGeneri(".btn-check:checked")){
        addBadge("Aggiungi almeno un genere che ti piace!");
    }
    if (!minLen("trackid", 1)) {
        addBadge("Aggiungi la tua traccia preferita!");
    }

    if (minLen("email", 6) && checkMail("email") && minLen("nome", 3)
        && checkPassword() && minLen("artistid", 1) && minLen("trackid", 1) && testPass()) {
        addSuccess()
    }
    return false;
}

```

Figure 5: check() ha una serie di test: se non passano vengono aggiunti dei badge a schermo che notificano all'utente cosa non funziona

```

function minLen(id, n) {
    return document.getElementById(id).value.length >= n
}

```

Figure 6: minLen(id,n) dato un id controlla che la lunghezza del valore dell'elemento con tale id sia maggiore di un certo numero

```

function checkGeneri(classe){
    return document.querySelectorAll(classe).length > 0
}

```

Figure 7: checkGeneri(classe) controlla che l'utente abbia selezionato almeno un genere

```

function checkPassword() {
    password = document.getElementById("password").value;
    passwprdR = /^[?=.*[a-z]](?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,}$/i
    return passwprdR.test(password)
}

```

Figure 8: checkPassword() controlla che la password abbia almeno una lettera maiuscola, una minuscola, un numero, e che sia lunga almeno 8 caratteri grazie al regex

```

function checkMail(id) {
    email = document.getElementById(id).value;
    emailR = /^[(([^<>()\\]+,;:\\s@")+(\\.\\[^<>()\\]+,;:\\s@")+)*|(".+"))@((\\[[0-9]{1,3}\\.\\[0-9]{1,3}\\.\\[0-9]{1,3}\\])|(([a-zA-Z-0-9]+.)+[a-zA-Z]{2,}))$/i;
    return emailR.test(email)
}

```

Figure 9: checkMail() controllo della mail grazie al regex

```

function testPass() {
    password = document.getElementById('password').value;
    password_2 = document.getElementById('password_conf').value;

    if (password.value != password_2.value) {
        return false
    }
    return true
}

```

Figure 10: testPass() controlla che i due campi password siano uguali

```

function addBadge(text) {
    var alert = document.getElementById('alert');
    alert.style.display = "block";
    alert.innerHTML = alert.innerHTML + "<p>" + text + "</p>"

}

```

Figure 11: addBadge() rende visibile il div che contiene il badge di avvertimento e aggiunge il testo desiderato

```

function clearBadge() {

    var alert = document.getElementById('alert');
    alert.style.display = "none";
    innerHTML = alert.innerHTML = ""

    var success = document.getElementById('success');
    success.style.display = "none";
    innerHTML = success.innerHTML = ""

}

```

Figure 12: clearBadge() nasconde i badge

```
function findUser(users, user) {
    for (var i = 0; i < users.length; i++) {
        if (users[i].email == user.email)
            return true
    }
    return false
}
```

Figure 13: `findUser()` controlla se c'è un utente con la stessa email di quello passato in input

```
function addSuccess() {
    var success = document.getElementById('success');
    success.style.display = "block";
    success.innerHTML = "<p> Utente registrato con successo! Per accedere a GCMO clicca in basso a su Accedi e inserisci le tue credenziali. </p>";
}
```

Figure 14: `addSuccess()` rende visibile il div che contiene il badge di successo

Per la selezione delle preferenze musicali bisogna interagire con le API di Spotify, per questo serve un access token, che verrà usato in tutte le pagine dell'applicazione in cui bisogna cercare o visualizzare informazioni su tracce, album e artisti. Le preferenze musicali servono per suggerire tracce all'utente in base ai suoi gusti.

```
function getToken() {  
  
    var url = "https://accounts.spotify.com/api/token"  
    var spotifyToken = window.localStorage.getItem("spotifyToken");  
  
    if (window.navigator.onLine != true) {  
        alert("Nessuna connessione a Internet")  
    } else {  
        fetch(url, {  
            method: "POST",  
            headers: {  
                "Content-Type": "application/x-www-form-urlencoded",  
                "Authorization": "Basic " + btoa(client_id + ":" + client_secret),  
            },  
            body: new URLSearchParams({ grant_type: "client_credentials" }),  
        })  
            .then((response) => {  
                if (response.ok) {  
                    return response.json()  
                }  
                throw new Error("Qualcosa è andato storto")  
            })  
            .then(tokenResponse => {  
                window.localStorage.setItem("spotifyToken", tokenResponse.access_token)  
                spotifyToken = window.localStorage.getItem("spotifyToken")  
            })  
            .catch((error) => console.log(error))  
  
        return spotifyToken;  
    }  
}
```

Figure 15: getToken()

```

function searchArtist(query) {
    var url = "https://api.spotify.com/v1/search?q=" + query + "&type=artist&limit=5"

    return fetch(url, {
        headers: {
            "Content-Type": "application/json",
            "Authorization": "Bearer " + window.localStorage.getItem("spotifyToken")
        },
    })
        .then((response) => {
            if (response.ok) {
                return response.json()
            }
            throw new Error("Qualcosa è andato storto")
        })
        .catch((error) => console.log(error))
}

```

Figure 16: searchArtist()

```

function searchBarArtist() {

    var tbody = document.getElementById('tbodyArtist')
    tbody.innerHTML = ""

    var query = document.getElementById('cercaArtista').value

    if (query.length > 2) {

        searchArtist(query)
            .then(json => {

                json.artists.items.forEach(artist => {

                    var generi = ""

                    if (artist.genres.length > 0) {
                        for (var i = 0; i < artist.genres.length; i++) {
                            generi += artist.genres[i] + "<br>"
                        }
                    }

                    tbody.innerHTML +=
                        `<tr><td><button class='btn btn-primary' onclick='setInputArtist('${artist.id}')'> + </button></td>` +
                        `+ "<td>" + artist.name + "</td><td>"` +
                        `+ generi + "</td><tr>"` +
                });
            });
    }
}

```

Figure 17: searchBarArtist()

```
function setInputArtist(id) {
    document.getElementById('artistid').value = id
}
```

Figure 18: setInputArtist()

```
function searchTrack(query) {
    var url = "https://api.spotify.com/v1/search?q=" + query + "&type=track&limit=5"

    return fetch(url, {
        headers: {
            "Content-Type": "application/json",
            "Authorization": "Bearer " + window.localStorage.getItem("spotifyToken")
        },
    })
        .then((response) => {
            if (response.ok) {
                return response.json()
            }
            throw new Error("Qualcosa è andato storto")
        })
        .catch((error) => console.log(error))
}
```

Figure 19: searchTrack()

```

function searchBarTrack() {
    var tbody = document.getElementById('tbodyTrack')
    tbody.innerHTML = ""

    var query = document.getElementById('cercaTraccia').value

    if (query.length > 2) {

        searchTrack(query)
            .then(json => {

                json.tracks.items.forEach(track => {

                    var artists = ""
                    var coverUrl = ""

                    if (track.album.images.length > 0) {
                        coverUrl = "<img src='"+ track.album.images[2].url + "'>"
                    }
                    else {
                        coverUrl = "foto non presente per la traccia corrispondente"
                    }

                    for (var i = 0; i < track.artists.length; i++) {
                        artists += track.artists[i].name + "<br>"
                    }

                    tbody.innerHTML +=
                        `<tr><td><button class='btn btn-primary' onclick='setInputTrack('${track.id}')'>+</button></td>` +
                        `+ "<td>" + track.name + "</td><td>"` +
                        `+ artists + "</td><td>"` +
                        `+ millisToMinutesAndSeconds(track.duration_ms) + "</td><td>"` +
                        `+ track.album.release_date + "</td><td>"` +
                        `+ coverUrl + "</td><tr>"`


                }));
            });
    }
}

```

Figure 20: searchBarTrack()

```

function setInputTrack(id) {
    document.getElementById('trackid').value = id
}

```

Figure 21: setInputTrack()

```

function getGenres() {
    var generi = document.getElementById('generi')
    generi.innerHTML = ""

    fetch(" https://api.spotify.com/v1/recommendations/available-genre-seeds", {
        headers: {
            "Content-Type": "application/json",
            "Authorization": "Bearer " + window.localStorage.getItem("spotifyToken")
        },
    })
        .then((response) => {
            if (response.ok) {
                return response.json()
            }
            throw new Error("Qualcosa è andato storto")
        })
        .then(json => {
            json.genres.forEach(genre => {

                generi.innerHTML += 
                    "<div class='mb-3 col'><input type='checkbox' class='btn-check' id='" + genre + "' autocomplete='off' value='" + genre + "'>" +
                    "<label class='btn btn-outline-primary' for='" + genre + "'>" + genre + "</label></div>"

            })
            registerLimitCheckEventListener()
        })
        .catch((error) => console.log(error))
}

```

Figure 22: getGeneri()

```

function limitCheck(event) {
    var max = 3;
    var checkedChecks = document.querySelectorAll(".btn-check:checked")
    if (checkedChecks.length >= max + 1) {
        alert("Puoi scegliere al massimo 3 generi preferiti. Deseleziona qualche altro genere per aggiungere quello che hai appena selezionato")
        event.target.checked = false
        return
    }
}

function registerLimitCheckEventListener() {
    const checkboxes = document.getElementsByClassName("btn-check")
    for (let i = 0; i < checkboxes.length; i++) {
        checkboxes[i].addEventListener("change", limitCheck)
    }
}

```

Figure 23: limitCheck() e registerLimitCheckEventListener() limitano la selezione dei generi a 3

Quando si clicca sul primo campo password appare un messaggio che si aggiorna man mano che i requisiti richiesti vengono soddisfatti. Il messaggio scompare quando si clicca fuori o su un altro campo.

The screenshot shows a user interface for entering a password. At the top, there is a text input field labeled "Password" containing the word "password". Below it is another text input field labeled "Conferma la password" also containing "password". A large gray box covers the middle section of the screen. Inside this box, the text "La password deve essere composta da:" is displayed in bold. Below this, there is a list of five items, each preceded by a red "X":

- X Una lettere minuscola**
- X Una lettere maiuscola**
- X Un numero**
- X Una lunghezza minima di 8 caratteri**

Figure 24: requisiti per la password

```

function showPasswordValidation() {

    var myInput = document.getElementById("password");
    var letter = document.getElementById("letter");
    var capital = document.getElementById("capital");
    var number = document.getElementById("number");
    var length = document.getElementById("length");

    myInput.onfocus = function () {
        document.getElementById("message").style.display = "block";
    }

    myInput.onblur = function () {
        document.getElementById("message").style.display = "none";
    }

    myInput.onkeyup = function () {

        var lowerCaseLetters = /[a-z]/g;
        if (myInput.value.match(lowerCaseLetters)) {
            letter.classList.remove("invalid");
            letter.classList.add("valid");
        } else {
            letter.classList.remove("valid");
            letter.classList.add("invalid");
        }

        var upperCaseLetters = /[A-Z]/g;
        if (myInput.value.match(upperCaseLetters)) {
            capital.classList.remove("invalid");
            capital.classList.add("valid");
        } else {
            capital.classList.remove("valid");
            capital.classList.add("invalid");
        }

        var numbers = /[0-9]/g;
        if (myInput.value.match(numbers)) {
            number.classList.remove("invalid");
            number.classList.add("valid");
        } else {
            number.classList.remove("valid");
            number.classList.add("invalid");
        }

        if (myInput.value.length >= 8) {
            length.classList.remove("invalid");
            length.classList.add("valid");
        } else {
            length.classList.remove("valid");
            length.classList.add("invalid");
        }
    }
}

```

Figure 25: showPasswordValidation() modifica lo stile del div messaggio che contiene i requisiti della password in base a quello che l'utente inserisce

```

<style>
    #message {
        display: none;
        background: #f1f1f1;
        color: #000;
        position: relative;
        padding: 20px;
        margin-top: 10px;
    }

    #message p {
        padding: 10px 35px;
        font-size: 16px;
    }

    .valid {
        color: green;
    }

    .valid:before {
        position: relative;
        left: -35px;
        content: "✓";
    }

    .invalid {
        color: red;
    }

    .invalid:before {
        position: relative;
        left: -35px;
        content: "✗";
    }
</style>

```

Figure 26: style del messaggio

Una volta che vengono caricati tutti i campi correttamente l'utente viene registrato e viene visualizzato un messaggio di successo.

Registrati a GCMO

Nome utente

Email address

Password

Conferma la password

Scegli un artista
3MZaB0qDrTJhTHQrO6Dq

add	nome	genere
<input checked="" type="checkbox"/>	Joji	viral pop
<input checked="" type="checkbox"/>	JoJo	dance pop hip hop pop post-teen pop r&b uk pop urban contemporary
<input checked="" type="checkbox"/>	JoJo Swa	
<input checked="" type="checkbox"/>	K-Ci & JoJo	contemporary r&b dance pop hip hop quiet storm r&b urban contemporary
<input checked="" type="checkbox"/>	Jojo's ASMR	asmr

Scegli i tuoi generi musicali preferiti (max 3)

acoustic	afrobeat	alt-rock	alternative	ambient	anime
black-metal	bluegrass	blues	bossanova	brazil	breakbeat
british	cantopop	chicago-house	children	chill	classical
club	comedy	country	dance	dancehall	death-metal
deep-house	detroit-techno	disco	disney	drum-and-bass	dub
dubstep	edm	electro	electronic	emo	folk
forro	french	funk	garage	german	gospel
goth	grindcore	groove	grunge	guitar	happy
hard-rock	hardcore	hardstyle	heavy-metal	hip-hop	holidays
honky-tonk	house	idm	indian	indie	indie-pop
industrial	iranian	j-dance	j-idol	j-pop	j-rock
jazz	k-pop	kids	latin	latino	maliy
mandopop	metal	metal-misc	metalcore	minimal-techno	movies
mpb	new-age	new-release	opera	papode	party
philippines-ops	piano	pop	pop-film	post-dubstep	power-pop
progressive-house	psych-rock	punk	punk-rock	r-n-b	rainy-day
reggae	reggaeton	road-trip	rock	rock-n-roll	rockabilly
romance	sad	salsa	samba	sertanejo	show-tunes
singer-songwriter	ska	sleep	songwriter	soul	soundtracks
spanish	study	summer	swedish	synth-pop	tango
techno	trance	trip-hop	turkish	work-out	world-music

Scegli una canzone

add	titolo	artista	durata	data di pubblicazione	cover
<input checked="" type="checkbox"/>	The Middle	Jimmy Eat World	2:46	2001-07-17	
<input checked="" type="checkbox"/>	The Spins	Mac Miller	3:16	2010-08-13	
<input checked="" type="checkbox"/>	The Middle	Zedd Maren Morris Grey	3:05	2018-01-23	
<input checked="" type="checkbox"/>	MIDDLE OF THE NIGHT	Elley Duhé	3:04	2020-01-10	
<input checked="" type="checkbox"/>	Brooklyn. Friday. Love.	The Midnight	3:52	2022-09-09	

Utente registrato con successo! Per accedere a GCMO clicca in basso a su Accedi e inserisci le tue credenziali.

[Registrati](#) oppure [Accedi](#)

Figure 27: schermata di registrazione con successo

```
▼ [ {nickname: "Luke Skywalker", email: "luke.skywalker@gmail.com", password: "Anewhope4", ...}, ...]
  ▶ 0: {nickname: "Luke Skywalker", email: "luke.skywalker@gmail.com", password: "Anewhope4", ...}
  ▶ 1: {nickname: "SpiderMan", email: "spider.man@hotmail.com", password: "farfromHome1", ...}
  ▶ 2: {nickname: "gretaVi", email: "greta.vigano97@gmail.com", password: "Settembre22", ...}
    email: "greta.vigano97@gmail.com"
    favArtist: "3MzsBdqRRTJihTHQr06Dq"
    favGenres: "emo%2Chardcore%2Cpunk"
    favTrack: "6GG73Jik4jUlQCKKg9JuG0"
    nickname: "gretaVi"
    password: "Settembre22"
```

Figure 28: array degli utenti nel local storage dopo la registrazione di "gretaVi"

3 Accesso

The screenshot shows a simple login interface. At the top right is the text "Accedi". Below it are two input fields: one for "Email address" containing "name@example.com" and another for "Password" containing "password". At the bottom left is a blue rectangular button with the text "Accedi" in white, and to its right, the text "oppure [Registrati](#)".

Figure 29: schermata di accesso

L'accesso deve essere verificato in ogni pagina, infatti se nessun utente è loggato non si potranno visualizzare i contenuti della pagina ma la schermata in figura-29.

```
function login() {
    users = window.localStorage.getItem('users')
    user = {
        email: document.getElementById('email').value,
        password: document.getElementById('password').value,
    }

    if (users == null) {
        users = []
    } else {
        users = JSON.parse(users)
    }
    if (users.find(u => u.email == user.email && u.password == user.password) == undefined) {
        alert("email e/o password errati")
    } else {
        window.localStorage.setItem('logged', user.email)
        logIn()
    }
}
```

Figure 30: login() controlla che la mail e la password inserite siano contenute nel local storage, in caso positivo salva nel local storage la mail dell'utente appena loggato e chiama la funzione logIn()

```
function logIn() {
    form.classList.add("visually-hidden")
    loggato.classList.remove("visually-hidden")
}
```

Figure 31: logIn() rende visibile il contenuto della pagina e nasconde il form di accesso

In ogni pagina è presente la funzione checkLogin() per verificare che ci sia un utente loggato.

```
function checkLogin() {
    if (window.localStorage.getItem('logged') != undefined) {
        logIn()
    } else {
        logOut()
    }
}
```

Figure 32: checkLogin()

Una volta inserite mail e password corrette e aver cliccato su accedi si potrà visualizzare la homepage.

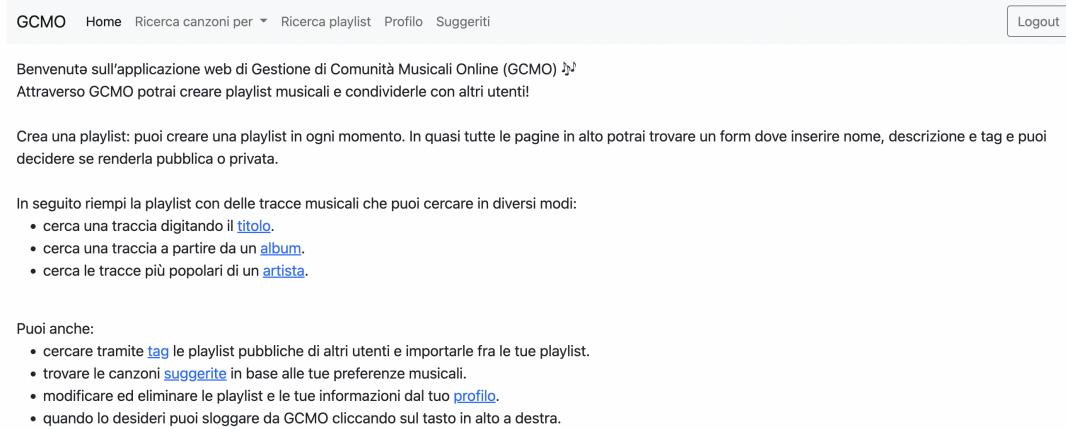


Figure 33: la homepage presenta una panoramica sulle funzionalità della applicazione

In ogni pagina è possibile effettuare il log out cliccando sul bottone "logout" nella navbar in alto a destra.

```
function logOut() {  
    window.localStorage.removeItem('logged')  
    form.classList.remove("visually-hidden")  
    loggato.classList.add("visually-hidden")  
}
```

Figure 34: logout()

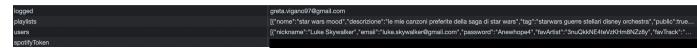


Figure 35: dopo il login di "gretaVi" nel local storage appare la variabile loggato

4 Creare una playlist

In quasi tutte le pagine è possibile creare una playlist compilando il form che si trova appena sotto la navbar.

The screenshot shows a web page with a header containing 'GCMO', 'Home', 'Ricerca canzoni per ▾', 'Ricerca playlist', and 'Profilo'. On the right is a 'Logout' button. Below the header is a form with the following fields:

- A text input field labeled 'dai un nome alla playlist'.
- A text input field labeled 'inserisci una descrizione della playlist'.
- A text input field labeled 'inserisci dei #tag che descrivono la playlist'.
- A radio button labeled 'pubblica' (selected) and a radio button labeled 'privata'.
- A blue 'crea' button.

Figure 36: form per la creazione di una playlist

il bottone "crea" chiama la funzione addplaylist().

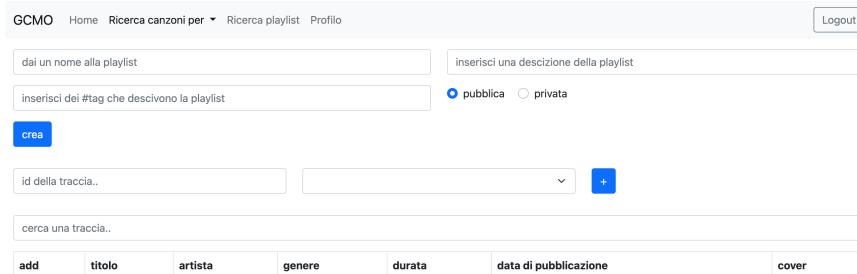
```
function addplaylist() {  
    playlists = window.localStorage.getItem('playlists')  
  
    if (playlists === null) {  
        playlists = []  
    } else {  
        playlists = JSON.parse(playlists)  
    }  
  
    var tracks = []  
  
    var playlist = {  
        nome: document.getElementById('playlist_nome').value,  
        descrizione: document.getElementById('playlist_desc').value,  
        tag: document.getElementById('playlist_tag').value,  
        public: document.getElementById('pubblica').checked,  
        owner: window.localStorage.getItem('logged'),  
        tracks: tracks  
    }  
  
    //controllo se la playlist inserita ha lo stesso nome di una playlist già creata  
    if (findPlaylist(playlists, playlist)) {  
        alert("playlist già registrata")  
    } else if(playlist.nome=="" || playlist.descrizione=="" || playlist.tag==""){  
        alert("hai lasciato uno o più campi vuoti, per creare una playlist devi completarli tutti")  
    }else{  
        playlists.push(playlist)  
        alert("playlist creata con successo!")  
    }  
  
    window.localStorage.setItem('playlists', JSON.stringify(playlists))  
}
```

Figure 37: addPlaylist() crea una playlist con le informazioni inserite dall'utente

5 Cercare tracce

```
function showPlaylistsOption() {  
    var sbody = document.getElementById('playlist_select')  
    sbody.innerHTML = ""  
  
    var playlists = window.localStorage.getItem('playlists')  
  
    if (playlists === null) {  
        playlists = []  
    } else {  
        playlists = JSON.parse(playlists)  
    }  
  
    for (var i = 0; i < playlists.length; i++) {  
        if (playlists[i].owner === window.localStorage.getItem('logged')) {  
            sbody.innerHTML += "<option value='"+playlists[i].nome+"'>"+ playlists[i].nome + "</option>"  
        }  
    }  
}
```

Figure 38: showPlaylistOption() aggiorna il select con le playlist che l'utente ha creato



The screenshot shows a web application interface for managing playlists. At the top, there's a navigation bar with links for 'Home', 'Ricerca canzoni per', 'Ricerca playlist', and 'Profilo'. On the right side of the header is a 'Logout' button. Below the header, there are several input fields and buttons:

- A text input field labeled "dai un nome alla playlist".
- A text input field labeled "inserisci una descrizione della playlist".
- A text input field labeled "inserisci dei #tag che descrivono la playlist".
- Two radio buttons for "pubblica" (public) and "privata" (private), with "pubblica" selected.
- A blue "crea" button.
- An input field for "id della traccia..".
- A dropdown menu.
- A small blue "+" button.
- An input field for "cerca una traccia..".
- A row of filters with labels: "add", "titolo", "artista", "genere", "durata", "data di pubblicazione", and "cover".

Figure 39: schermata per la ricerca di una traccia in base al suo nome

Figure 40: la tabella dei risultati della ricerca viene aggiornata ogni volta che l’utente digita (onkeyup) nella barra di ricerca

```
function search(query) {
    var url = "https://api.spotify.com/v1/search?q=" + query + "&type=track&limit=15"

    return fetch(url, {
        headers: {
            "Content-Type": "application/json",
            "Authorization": "Bearer " + window.localStorage.getItem("spotifyToken")
        },
    })
        .then((response) => {
            if (response.ok) {
                return response.json()
            }
            throw new Error("Qualcosa è andato storto")
        })
        .catch((error) => console.log(error))
}
```

Figure 41: search(query) esegue una chiamata all’API di Spotify

```

function searchBar() {

    var tbody = document.getElementById('tbody')
    tbody.innerHTML = ""

    var query = document.getElementById('cerca').value

    if (query.length > 2) {
        let tracks = search(query).then(json => json.tracks.items)
        const createRow = track => {

            var artists = ""
            var genere = ""
            var coverUrl = ""

            if (track.album.images.length > 0) {
                coverUrl = "<img src='" + track.album.images[2].url + "'>"
            }
            else {
                coverUrl = "foto non presente per la traccia corrispondente"
            }

            for (var i = 0; i < track.artists.length; i++) {
                artists += track.artists[i].name + "<br>"
                getArtistGenres(track.artists[i].id).then(response => {
                    genere += response + "<br>"

                    tbody.innerHTML +=
                        '<tr><td><button class="btn btn-primary" onclick="setInput("${track.id}")"> + </button></td>' +
                        '<td>' + track.name + "</td><td>" +
                        artists + "</td><td>" +
                        genere + "</td><td>" +
                        millisToMinutesAndSeconds(track.duration_ms) + "</td><td>" +
                        track.album.release_date + "</td><td>" +
                        coverUrl + "</td><tr>'

                })
            }
            //per ciascun track viene creata una row
        tracks.then(r => r.map(track => createRow(track)))
    }
}

```

Figure 42: searchBar() popola la tabella dei risultati in base a quello che l'utente ha inserito nella barra di ricerca

```

function getArtistGenres(id) {
    return fetch("https://api.spotify.com/v1/artists/" + id, {
        headers: {
            "Content-Type": "application/json",
            "Authorization": "Bearer " + window.localStorage.getItem("spotifyToken")
        },
    })
        .then((response) => {
            if (response.ok) {
                return response.json()
            }
            throw new Error("Qualcosa è andato storto")
        })
        .then(json => {
            var generi = ""

            if (json.genres.length > 0) {
                for (var i = 0; i < json.genres.length; i++) {
                    generi += json.genres[i] + "<br>"
                }
            }

            return generi
        })
        .catch((error) => console.log(error))
}

```

Figure 43: getArtistGenres() ritorna i generi degli artisti

add	titolo	artista	genere	durata	data di pubblicazione	cover
	All Star	Smash Mouth	alternative rock pop rock	3:20	1999-06-08	
	All Girls Are The Same	Juice WRLD	chicago rap melodic rap	2:46	2018-12-10	
	All The Stars (with SZA)	Kendrick Lamar SZA	conscious hip hop hip hop	3:52	2018-02-09	

Figure 44: quando si trova la traccia desiderata l'utente deve cliccare sul tasto "+" e l'id della traccia verrà copiato nell'apposito campo. In seguito l'utente dovrà selezionare la playlist e cliccare su "+" per aggiungere la traccia alla playlist

```

function add2playlist() {

    var trackid = document.getElementById('trackid').value
    var selected = document.getElementById('playlist_select').value

    var playlists = window.localStorage.getItem('playlists')

    if (playlists != null) {
        playlists = JSON.parse(playlists)
        for (var i = 0; i < playlists.length; i++) {
            if (playlists[i].nome == selected) {
                if(playlists[i].tracks.includes(trackid)){
                    alert("traccia già presente nella playlist")
                }else{
                    playlists[i].tracks.push(trackid)
                    alert("traccia aggiunta alla playlist con successo")
                    window.localStorage.setItem('playlists', JSON.stringify(playlists));
                }
            }
        }
    }
}

```

Figure 45: add2playlist() aggiunge la traccia alla playlist

```

▼ [{nome: "star wars mood", descrizione: "le mie canzoni preferite della saga di star wars",...}
  ▶ 0: {nome: "star wars mood", descrizione: "le mie canzoni preferite della saga di star wars",...}
  ▶ 1: {nome: "gaming", descrizione: "tracce da ascoltare mentre si gioca",...}
  ▶ 2: {nome: "homecoming", descrizione: "canzoni da ascoltare in viaggio", tag: "chill rap energy morning ",...}
  ▶ 3: {nome: "ambient", descrizione: "canzoni per rilassarsi", tag: "ambient natura folk pioggia",...}
  ▶ 4: {nome: "car trip", descrizione: "canzoni da ascoltare e cantare in macchina ",...
      descrizione: "canzoni da ascoltare e cantare in macchina "
      nome: "car trip"
      owner: "greta.vigano97@gmail.com"
      public: true
      tag: "macchina trip happy car "
      tracks: ["3cfd0d40hv2snfaKAnMdnvK"]}

```

Figure 46: array delle playlist nel local storage dopo che l'utente loggato "gretaVi" ha creato la playlist "car trip" e aver aggiunto una traccia

6 Cercare album e artisti

Il funzionamento per la ricerca di album è molto simile a quello per la ricerca di artisti. Per visualizzare le tracce di un album o le tracce più ascoltate di un artista si clicca su un bottone che porta a un'altra pagina, passando l'id dell'album o dell'artista nell'url.

more info	titolo	artista	tracce totali	data di pubblicazione	cover
get album tracks	The Wall	Pink Floyd	26	1979-11-30	
get album tracks	Dangerous: The Double Album	Morgan Wallen	30	2021-01-08	
get album tracks	The Way It Is	Keyshia Cole	12	2005-01-01	
get album tracks	Watch The Throne (Deluxe)	Jay-Z & Kanye West	18	2011-08-08	

Figure 47: schermata di ricerca di un album

```

function searchBar() {
    var tbody = document.getElementById('tbody')
    tbody.innerHTML = ""

    var query = document.getElementById('cerca').value
    if (query.length > 2) {
        search(query)
            .then(json => {
                json.albums.items.forEach(album => {
                    var artists = ""
                    var coverUrl = ""

                    if (album.artists.length > 0) {
                        for (var i = 0; i < album.artists.length; i++) {
                            artists += album.artists[i].name + "<br>"
                        }
                    }

                    if(album.images.length>0){
                        coverUrl = "<img src='"+album.images[2].url+">"
                    }
                    else{
                        coverUrl = "foto non presente per l'album corrispondente"
                    }

                    tbody.innerHTML +=
                    "<tr><td><button class='btn btn-primary' onclick='window.location.href='albumTracks.html?url=${album.id}'>get album tracks</button></td>" +
                    "<td>" + album.name + "</td><td>" +
                    artists + "</td><td>" +
                    album.total_tracks + "</td><td>" +
                    album.release_date + "</td><td>" +
                    coverUrl + "</td><tr>" +
                });
            })
    }
}

```

Figure 48: searchBar() per gli album

The screenshot shows the GCMO application's interface for creating a new playlist. At the top, there is a navigation bar with links for 'Home', 'Ricerca canzoni per', 'Ricerca playlist', 'Profilo', and 'Suggeriti'. On the right side of the header is a 'Logout' button. Below the header, there are several input fields and buttons:

- A text input field labeled 'dai un nome alla playlist'.
- A text input field labeled 'inserisci una descrizione della playlist'.
- A dropdown menu with options 'pubblica' (selected) and 'privata'.
- A blue 'crea' button.
- An input field containing the string '62zuXt6X5B6YQOBNFV2eXG'.
- A dropdown menu showing the value 'car trip'.
- A small blue '+' button.

Below these controls, the title 'Tracce dell'album' is displayed above a table. The table has the following columns: 'add', 'titolo', 'artista', and 'durata'. It lists five tracks from the album 'The Wall' by Pink Floyd:

add	titolo	artista	durata
	In the Flesh?	Pink Floyd	3:18
	The Thin Ice	Pink Floyd	2:26
	Another Brick in the Wall, Pt. 1	Pink Floyd	3:12
	The Happiest Days of Our Lives	Pink Floyd	1:51
	Another Brick in the Wall, Pt. 2	Pink Floyd	3:59

Figure 49: quando viene cliccato "get album tracks" su un album si passa a questa schermata, dove si possono visualizzare le informazioni delle tracce contenute nell'album stesso

```

function getAlbumTracks() {
    var id = getParameter('url');

    fetch("https://api.spotify.com/v1/albums/" + id + "/tracks", {
        headers: {
            "Content-Type": "application/json",
            "Authorization": "Bearer " + window.localStorage.getItem("spotifyToken")
        },
    })

    .then((response) => {
        if (response.ok) {
            return response.json()
        }
        throw new Error("Qualcosa è andato storto")
    })
    .then(json => {
        json.items.forEach(track => {
            var artists = ""

            for (var i = 0; i < track.artists.length; i++) {
                artists += track.artists[i].name + "<br>"
            }

            tbody.innerHTML +=
                `<tr><td><button class='btn btn-primary' onclick='setInput('${track.id}')'>+</button></td>` +
                `+ "<td>" + track.name + "</td><td>"` +
                `+ artists + "</td><td>"` +
                `+ millisToMinutesAndSeconds(track.duration_ms) + "</td><tr>"` +
            );
        })
        .catch((error) => console.log(error))
    })
}

```

Figure 50: getAlbumTracks()

7 Cercare playlist

Per cercare playlist di altri utenti basta digitare una parola chiave -un tag- nella barra di ricerca.

The screenshot shows a search interface for playlists. At the top, there is a navigation bar with links: GCMO, Home, Ricerca canzoni per, Ricerca playlist, Profilo, Suggeriti, and Logout. Below the navigation bar are three input fields: 'dai un nome alla playlist', 'inserisci una descrizione della playlist', and 'inserisci dei #tag che descrivono la playlist'. There are also two radio buttons for 'pubblica' (selected) and 'privata'. A blue 'crea' button is located below these fields. Below the input fields is a search bar containing the text 'st'. The main area displays a table of search results:

import	titolo	descrizione	tag	tracks	durata totale
	star wars mood	le mie canzoni preferite della saga di star wars	starwars guerre stellari disney orchestra	pubblica	Star Wars (Main Theme) Imperial Attack Star Wars: Princess Leia 16:11

Figure 51: schermata di ricerca di playlist in base ai tag

```

function searchTag() {
    var tbody = document.getElementById('resultPlaylist')
    tbody.innerHTML = ""

    var playlists = window.localStorage.getItem('playlists')
    var query = document.getElementById('cerca').value

    if (query.length > 1) {
        if (playlists != null) {
            playlists = JSON.parse(playlists)
            for (var i = 0; i < playlists.length; i++) {
                if (playlists[i].owner != window.localStorage.getItem('logged') && playlists[i].public == true
                    && playlists[i].tag.toLowerCase().includes(query.toLowerCase())){

                    let playlistName = playlists[i].name
                    let playlistDescription = playlists[i].descrizione
                    let playlistTag = playlists[i].tag
                    let playlistPublic = playlists[i].public
                    let trackInfo = playlists[i].tracks.map(track => getTracksInfo(track))

                    tbody.innerHTML +=
                        `<tr><td><button class='btn btn-primary' onclick='importPlaylist("${playlistName}")'> + </button></td>` +
                        `<td>${playlistName}</td><td>${
                            playlistDescription +
                            playlistTag +
                            stato(playlistPublic) +
                            "</td><td id='tracks-' + playlistName + ''></td>" +
                            `<td id='time-' + playlistName + ''> 00:00 </td><tr>`}`

                    const trackInfoResponse = trackInfo.map(promise => promise.then(response => {
                        let tracksTr = document.getElementById("tracks-" + playlistName)
                        tracksTr.innerHTML += response.name + "<br>"
                        let timeTr = document.getElementById("time-" + playlistName)
                        var contenuto = minutesAndSecondsTomillis(timeTr.innerHTML)
                        var nuovo = response.duration_ms
                        timeTr.innerHTML = millisToMinutesAndSeconds(nuovo + contenuto)

                    }))

                }
            }
        }
    }
}

```

Figure 52: searchTag() permette la ricerca di playlist in base al tag

```

function importPlaylist(nome) {
    clearBadgeImportSuccess()

    var playlists = window.localStorage.getItem('playlists')
    playlists = JSON.parse(playlists)

    for (var i = 0; i < playlists.length; i++) {
        if (playlists[i].nome == nome && playlists[i].owner != window.localStorage.getItem('logged'))
            var index = i
    }

    var playlist = {
        nome: nome,
        descrizione: playlists[index].descrizione,
        tag: playlists[index].tag,
        public: playlists[index].public,
        owner: window.localStorage.getItem('logged'),
        tracks: playlists[index].tracks
    }

    //controllo se la playlist inserita ha lo stesso nome di una playlist già creata
    if (findPlaylist(playlists, playlist)) {
        alert("playlist già registrata")
    } else {
        playlists.push(playlist)
        importSuccess()
    }

    window.localStorage.setItem('playlists', JSON.stringify(playlists))
}

```

Figure 53: importPlaylist() copia la playlist che l'utente loggato vuole importare e la aggiunge alle sue playlist

```

[{"nome: "star wars mood", descrizione: "le mie canzoni preferite della saga di star wars"}, {"nome: "star wars mood", descrizione: "le mie canzoni preferite della saga di star wars"}, {"nome: "gaming", descrizione: "tracce da ascoltare mentre si gioca"}, {"nome: "homecoming", descrizione: "canzoni da ascoltare in viaggio", tag: "chill rap energy morning"}, {"nome: "ambient", descrizione: "canzoni per rilassarsi", tag: "ambient natura folk pioggia"}, {"nome: "car trip", descrizione: "canzoni da ascoltare e cantare in macchina", tag: "canzoni da ascoltare e cantare in macchina"}, {"nome: "car trip", owner: "greta.vigano97@gmail.com", public: true, tag: "macchina trip happy car"}, {"tracks: ["f3cf0d40MvZsnFaKAnMdvnK"]}

```

Figure 54: array delle playlist nel local storage dopo che l'utente loggato "gretaVi" ha copiato la playlist "star wars mood". Ora sono presenti due playlist con stesso nome e stesse tracce, ma con proprietari diversi.

8 Tracce suggerite

In questa pagina è possibile trovare tracce suggerite in base alle preferenze musicali dell’utente.

The screenshot shows a user interface for creating a playlist. At the top, there are input fields for the playlist name ('dai un nome alla playlist') and description ('inserisci una descrizione della playlist'), and options for public or private status ('pubblica' or 'privata'). Below these are buttons for 'crea' (create) and 'id della traccia...' (track ID). A search bar contains the text 'car trip'. The main area displays a table titled 'Tracce suggerite in base alle tue preferenze musicali' (Suggested tracks based on your musical preferences). The table has columns: add (+), titolo (title), artista (artist), genere (genre), durata (duration), data di pubblicazione (publication date), and cover (cover art). Two tracks are listed:

add	titolo	artista	genere	durata	data di pubblicazione	cover
+	Revolution	Authority Zero	arizona punk melodic hardcore punk ska punk skate punk	2:25	2004-06-29	
+	The Art Of Losing	American Hi-Fi	pop punk pop rock skate punk	3:23	2003-02-24	

Figure 55: schermata con le tracce suggerite

```
function getRecommendations() {
    var tbody = document.getElementById('tbody')
    tbody.innerHTML = ""

    var users = window.localStorage.getItem('users')
    users = JSON.parse(users)

    for (var i = 0; i < users.length; i++) {
        if (users[i].email == window.localStorage.getItem('logged')) {
            var index = i
        }
    }

    var url = "https://api.spotify.com/v1/recommendations?limit=10&market=IT&seed_artists=" + users[index].favArtist + "&seed_genres=" + users[index].favGenres + "&seed_tracks=" + users[index].favTrack

    return fetch(url, {
        headers: {
            "Content-Type": "application/json",
            "Authorization": "Bearer " + window.localStorage.getItem("spotifyToken")
        },
    })
        .then((response) => {
            if (response.ok) {
                return response.json()
            }
            throw new Error("Qualcosa è andato storto")
        })
        .catch((error) => console.log(error))
}
```

Figure 56: getRecommendations() esegue una chiamata all’API per le tracce suggerite

```

function showRecommendations(){
  let tracks = getRecommendations().then(json => json.tracks)
  const createRow = track => {

    var artists = ""
    var genre = ""
    var coverUrl = ""

    if (track.album.images.length > 0) {
      coverUrl = "<img src='" + track.album.images[2].url + "'>"
    }
    else {
      coverUrl = "foto non presente per la traccia corrispondente"
    }

    for (var i = 0; i < track.artists.length; i++) {
      artists += track.artists[i].name + "<br>"
      getArtistGenres(track.artists[i].id).then(response => {

        genre += response + "<br>

        tbody.innerHTML +=
          `<tr><td><button class='btn btn-primary' onclick='setInput('${track.id}')'> + </button></td>` +
          `+ <td>${track.name}</td><td>` +
          `+ ${artists}</td><td>` +
          `+ ${genre}</td><td>` +
          `+ millisToMinutesAndSeconds(track.duration_ms) + </td><td>` +
          `+ track.album.release_date + </td><td>` +
          `+ ${coverUrl} + </td><tr>` +
        `)>
      )
    }
  }
  tracks.then(r => r.map(track => createRow(track)))
  //per ciascun track create row
}

```

Figure 57: showRecommendations() è quasi identico a searchBar() per le tracce, senza la barra di ricerca

9 Profilo

In questa pagina è possibile modificare ed eliminare le informazioni legate a playlist e utente.

The screenshot shows a web application interface for managing a user's profile. At the top, there is a navigation bar with links for 'Home', 'Ricerca canzoni per', 'Ricerca playlist', 'Profilo' (which is the active tab), and 'Suggeriti'. On the right side of the header is a 'Logout' button. Below the header, there is a form for creating a new playlist. It includes fields for 'dai un nome alla playlist' (enter a name for the playlist), 'inserisci una descrizione della playlist' (enter a description for the playlist), and 'inserisci dei #tag che descrivono la playlist' (enter tags that describe the playlist). There are also radio buttons for 'pubblica' (public) and 'privata' (private), with 'pubblica' selected. A blue 'crea' (create) button is located below the form. At the bottom of the page, there are two buttons: 'modifica le info del tuo account' (modify your account info) and 'elimina il tuo account' (delete your account). The main content area is titled 'Le mie playlist' (My playlists) and displays a table with two rows of data. The columns are: modifica (edit), titolo (title), descrizione (description), tag (tags), stato (status), tracks (tracks), durata totale (total duration), and elimina (delete). The first playlist is titled 'car trip' with description 'canzoni da ascoltare e cantare in macchina', tags 'macchina trip happy car', status 'pubblica', tracks 'All Star', duration '3:20', and a delete button. The second playlist is titled 'star wars mood' with description 'le mie canzoni preferite della saga di star wars', tags 'starwars guerre stellari disney orchestra', status 'pubblica', tracks 'Star Wars: Princess Leia Star Wars (Main Theme) Imperial Attack', duration '16:11', and a delete button.

modifica	titolo	descrizione	tag	stato	tracks	durata totale	elimina
...	car trip	canzoni da ascoltare e cantare in macchina	macchina trip happy car	pubblica	All Star	3:20	x
...	star wars mood	le mie canzoni preferite della saga di star wars	starwars guerre stellari disney orchestra	pubblica	Star Wars: Princess Leia Star Wars (Main Theme) Imperial Attack	16:11	x

Figure 58: schermata del profilo

In questa pagina si può direttamente eliminare profilo e playlist.

```
function deleteProfile() {  
  
    var users = window.localStorage.getItem('users')  
    users = JSON.parse(users)  
  
    for (var i = 0; i < users.length; i++) {  
        if (users[i].email == window.localStorage.getItem('logged'))  
            var index = i  
    }  
  
    users.splice(index, 1)  
    localStorage.setItem('users', JSON.stringify(users))  
    logOut()  
}
```

Figure 59: deleteProfile() elimina il profilo loggato e esegue logout dall'applicazione

```
function deletePlaylist(index) {
    var playlists = window.localStorage.getItem('playlists')
    playlists = JSON.parse(playlists)
    playlists.splice(index, 1)
    localStorage.setItem('playlists', JSON.stringify(playlists))
    window.location.reload()
}
```

Figure 60: deletePlaylist() elimina la playlist selezionata

9.1 Modifica playlist

I campi di testo sono già compilati con le informazioni che l'utente ha inserito al momento della creazione della playlist (o dell'ultima modifica).

The screenshot shows a web interface for modifying a playlist. At the top, there's a navigation bar with links: GCMO, Home, Ricerca canzoni per (with a dropdown arrow), Ricerca playlist, Profilo, Suggeriti, and a Logout button. Below the navigation, there are input fields for the playlist's name ('car trip'), description ('canzoni da ascoltare e cantare in macchina'), and tags ('macchina trip happy car'). A radio button indicates the playlist is publica (public). An 'Aggiorna' (Update) button is visible. Below these fields is a section titled 'Canzoni della playlist' containing a table with one row of data.

elimina	titolo	artista	durata	data di pubblicazione	cover
	All Star	Smash Mouth	3:20	1999-06-08	

Figure 61: schermata di modifica di una playlist

```

function updatePlaylist() {
    var playlists = window.localStorage.getItem('playlists')
    var playlists = JSON.parse(playlists)

    for (var i = 0; i < playlists.length; i++) {
        if (playlists[i].nome == idPlaylist)
            var index = i
    }

    var playlist = {
        nome: document.getElementById('playlist_nome').value,
        descrizione: document.getElementById('playlist_desc').value,
        tag: document.getElementById('playlist_tag').value,
        public: document.getElementById('pubblica').checked,
        owner: window.localStorage.getItem('logged'),
        tracks: playlists[index].tracks
    }

    for (var i = 0; i < playlists.length; i++) {
        if (playlists[i].nome == playlist.nome && i != index){
            alert("Non è possibile aggiornare la playlist perché esiste un'altra playlist con lo stesso nome")
            return
        }
    }
    if(playlist.nome=="" || playlist.descrizione=="" || playlist.tag==""){
        alert("hai lasciato uno o più campi vuoti, per creare una playlist devi completarli tutti")
        return
    }

    playlists[index] = playlist
    alert("playlist aggiornata con successo!")
    window.localStorage.setItem('playlists', JSON.stringify(playlists))
}

```

Figure 62: updatePlaylist() aggiorna la playlist selezionata

9.2 Modifica profilo

I campi di testo sono già compilati con le informazioni che l'utente ha inserito al momento della registrazione (o dell'ultima modifica).

The screenshot shows a web-based profile modification form. At the top, there's a navigation bar with links for 'Home', 'Ricerca canzoni per', 'Ricerca playlist', 'Profilo', and 'Suggeriti'. On the right side of the header is a 'Logout' button. Below the header, there are input fields for 'Nome utente' (gretaVi), 'Email address' (greta.vigano97@gmail.com), 'Password' (a masked password), and 'Conferma la password' (also a masked password). There's also a search bar for artists ('cerca un artista..') and a code field ('3MZsBdqDrRTjhTHQrO6Dq'). A section titled 'Scegli un artista' contains a table with columns 'add', 'nome', and 'genere'. Below this is a large grid of musical genres. At the bottom, there's another search bar for songs ('cerca una traccia..') and a code field ('6GG73Jik4JlJlQCkKg9JuGO'). A row of buttons at the bottom includes 'add', 'titolo', 'artista', 'durata', 'data di pubblicazione', 'cover', and an 'Aggiorna' button.

Figure 63: schermata di modifica del profilo

```

var idUsers = getParameter('url');

function fillInputs() {

    var users = window.localStorage.getItem('users')
    var users = JSON.parse(users)

    for (var i = 0; i < users.length; i++) {
        if (users[i].email == idUsers)
            var index = i
    }

    if (index != undefined) {

        document.getElementById('nome').value = users[index].nickname
        document.getElementById('password_conf').value = users[index].password
        document.getElementById('artistid').value = users[index].favArtist
        document.getElementById('trackid').value = users[index].favTrack
        document.getElementById('email_').value = users[index].email
        document.getElementById('password_').value = users[index].password
    }
    else {
        window.location.href = 'myProfile.html'
        return false;
    }
}

```

Figure 64: fillInputs() riempie i campi di testo con le info dell'utente

```

function updateUserInfo() {
    var users = window.localStorage.getItem('users')
    var users = JSON.parse(users)

    for (var i = 0; i < users.length; i++) {
        if (users[i].email == idUsers)
            var index = i
    }

    var genres = ""
    var checkedChecks = document.querySelectorAll(".btn-check:checked")
    for (var i = 0; i < checkedChecks.length; i++) {
        if (i == 0) {
            genres += checkedChecks[i].id
        } else {
            genres += "%2C" + checkedChecks[i].id
        }
    }

    user = {
        nickname: document.getElementById('nome').value,
        email: document.getElementById('email_').value,
        password: document.getElementById('password_').value,
        favArtist: document.getElementById('artistid').value,
        favTrack: document.getElementById('trackid').value,
        favGenres: genres
    }

    for (var i = 0; i < users.length; i++) {
        if (users[i].email == user.email && users[i].email!=idUsers)
            alert("Non è possibile aggiornare il profilo perché la mail inserita appartiene a un altro utente")
    }
    if (!minLen("email_", 6) || !checkMail("email_") || !minLen("nome", 3)
        || !testPass() || !checkPassword() || !minLen("artistid", 1) || !minLen("trackid", 1)) {
        alert("Non è possibile aggiornare il profilo perché uno dei campi è stato compilato in modo scorretto")
    }
    else {
        users[index] = user
        alert("profilo aggiornato con successo")
    }
    window.localStorage.setItem('users', JSON.stringify(users))
}

```

Figure 65: updateUserInfo() è molto simile a registrati()

```

function checkUp() {

    clearBadge()

    if (!minLen("email_", 6)) {
        addBadge("Email troppo corta!");
    }
    if (!checkMail("email_")) {
        addBadge("Email non valida!");
    }
    if (!minLen("nome", 3)) {
        addBadge("Nome troppo corto!");
    }
    if (!checkPassword()) {
        addBadge("La password non soddisfa i requisiti dei criteri di password");
    }
    if (!testPass()) {
        addBadge("La password inserita non coincide con la prima!");
    }
    if (!minLen("artistid", 1)) {
        addBadge("Aggiungi il tuo artista preferito!");
    }
    if (!minLen("trackid", 1)) {
        addBadge("Aggiungi il tuo artista preferito!");
    }

    return false;
}

```

Figure 66: checkUp() è molto simile a check()