**Pràctica 6: Disseny d’una aplicació web amb SQL Lite i implementació de mesures de securització d’entrada a formularis**

**Objectius:**

L’objectiu d’aquesta activitat és seguir una breu documentació tècnica per desplegar una petita aplicació web en local, amb node.js, HTML/Javascript i SQL Lite per efectuar proves sobre entrada de dades dels formularis que dissenyeu i proposar solucions per a millorar-ho.

L’objectiu no és que feu una pàgina web complexa, el que es vol és que genereu un mínim entorn de prova (sobre el qual tingueu control absolut) per dur a terme proves i dissenyar sobre el vostre codi mateix solucions que abordin 3 aspectes insegurs dels formularis (SQL injections, Cross Site Scripting i HTML injection). Es possible que s’utilitzi el codi per alguna pràctica de més endavant també.

Haureu de documentar el procés a través d’una memòria tècnica en PDF, per tant és aconsellable fer una lectura de tots els passos, organitzar un índex del qual s’ha de fer i anar documentant amb proves gràfiques tots els procediments realitzats. Una proposta d’índex seria:

1.Introducció i objectius (sigueu breus)

2.Guia del desplegament de la vostra aplicació

3.Proves de vulnerablitats del vostre formulari (aquí encara no heu d’aplicar cap mesura de securització de software)

3.1 Exemple de SQL Injection sezill (obtenció de dades amb la consulta SQL amb 1=1)

3.2. Algun exemple de XSS (la típica introducció del alert en el formulari o camp dins del HTML)

3.3. Alguna prova d'injecció de codi HTML

4.Implementació de mesures per prevenir les injeccions de codi anteriors.

5.Exemple de funcionament de les millores

Bibliografia i referencies utilitzades

**Resultat d’aprenentatge:**

Ra3. Detecta i corregeix vulnerabilitats d'aplicacions web analitzant el seu codi font i configurant servidors web.

**Agrupació:**

Individual

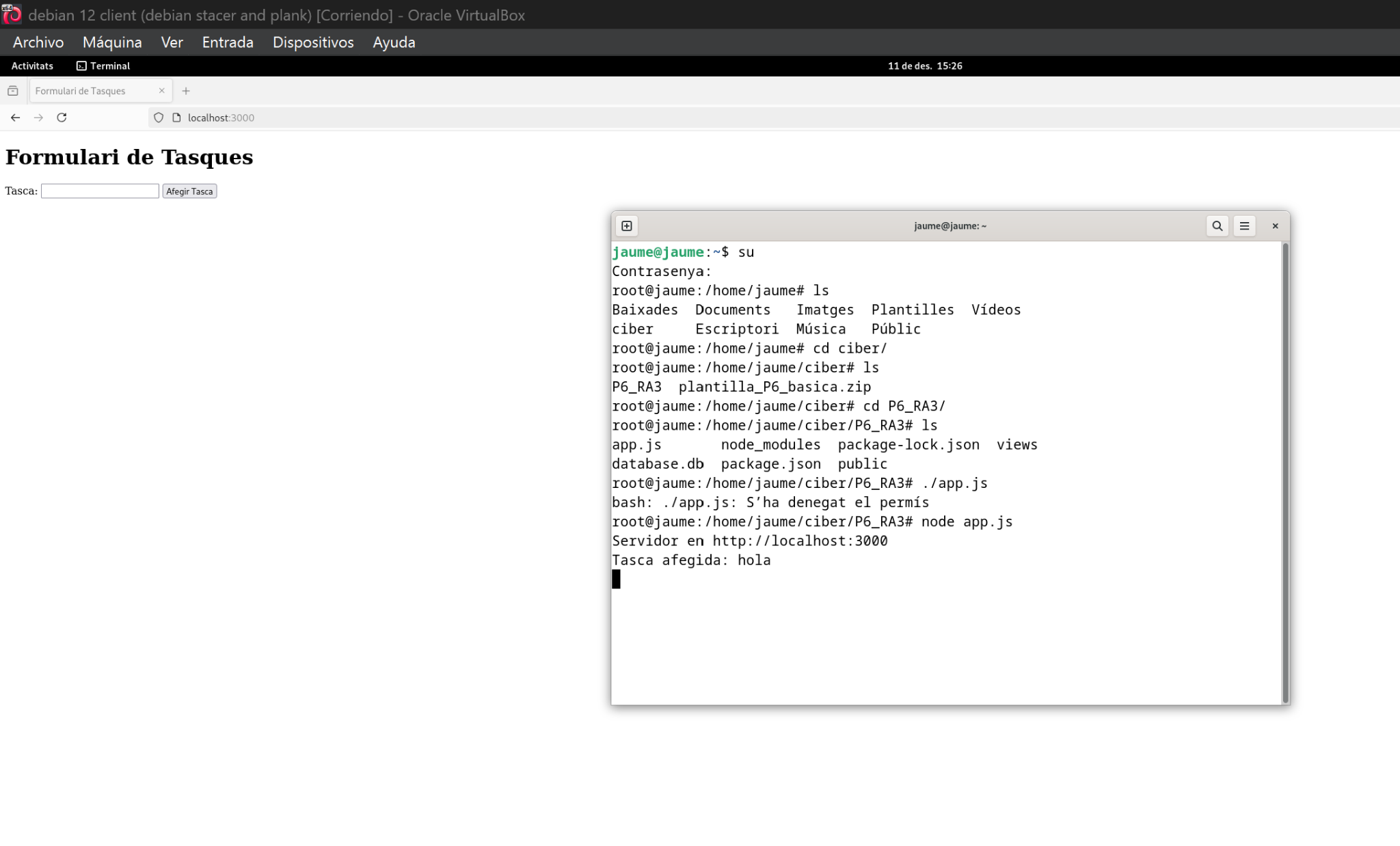
**Enunciat:**

Primer de tot feu una senzilla pàgina web amb un servidor node.js que gestioni les peticions en local i que tingui un petit servei SQL Lite que pugui gravar informacions senzilles dins del vostre formulari.

Per fer-ho podeu seguir la guia que us he preparat i es troba en un document annex amb el nom de “Guia desplegament app web amb node.js”.

Un cop desplegada l’aplicació podeu afegir algun atribut a la taula tasques i algun camp més que reculli aquestes dades des del formulari de la vostra web. Tot això es per fer un setting senzill de l’aplicació i començar des de 0.

Tal com el que es mostra en la següent captura:



A partir d’aquí introduïu com a mínim un loggin i password a la vostre mini app web (haureu de crear una taula com a mínim) per fer registre d'usuaris.

Introduïu elements HTML al vostre gust per poder provar les vulnerabilitats que es proposen (XXS, injecció SQL i HTML). Proveu la vulnerabilitat, adjunteu la captura a la vostra memòria amb el resultat (mostra la taula o contingut que no s’hauria de mostrar, i introduïu un element javascript i HTML que no s’hauria de produir). Per implementar mesures que

Documenteu bé a la memòria la solució que heu aplicat i el com ja no succeeix.

Recordeu intentar seguir l'índex proposat a la part d’objectius

# Introducción y objetivos.

Vamos a crear una página web mediante el framework ExpressJS. La página tratará sobre música clásica. A parte de la pasión que yo tengo por la música, otro motivo por el cual selecciono este tema, es porque muchas páginas que hablan sobre este tipo de música están totalmente desfasadas. Son páginas sin ningún tipo de mantenimiento y que además fueron creadas hace mucho tiempo por lo que están totalmente anticuadas.   
  
Vamos a mostrar ciertas vulnerabilidades web que son muy fáciles de ser solucionadas, que son también muy fáciles de que se nos olviden y que además son altamente peligrosas si no se toman en consideración porque se pueden hacer ciertas acciones ilegales si no se tiene una página en condiciones.

Esta página se va a plantear de una manera diferente. La empresa ClassicalFM ha lanzado una nueva iniciativa. Por tal de expandir más el conocimiento que disponen los oyentes de este estilo de música, esta empresa ha decidido de lanzar una página web la cual los usuarios van a depositar los conocimientos que tienen sobre este estilo de música. Para poder acceder a esta página deberán pagar una membresía mensual, la cual garantizará el acceso de estos para poder depositar sus conocimientos. Dicha membresía tendrá un coste de 13,99 euros al mes. Esta incluye acceso a la página ClassicalFM para poder ver sus artículos y también a la página para poder depositar sus conocimientos.

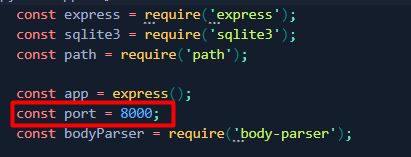
# Guía de despliegue de la aplicación

Para poder iniciar esta aplicación se deberá primeramente abrir un cmd o el propio visual studio. Una vez estemos dentro de la carpeta del proyecto la primera cosa a realizar es ejecutar el siguiente comando (Esto en el supuesto caso que no exista la carpeta node\_modules):

|  |
| --- |
| **npm install** |

A continuación, ejecutaremos el siguiente comando para inicializar la aplicación

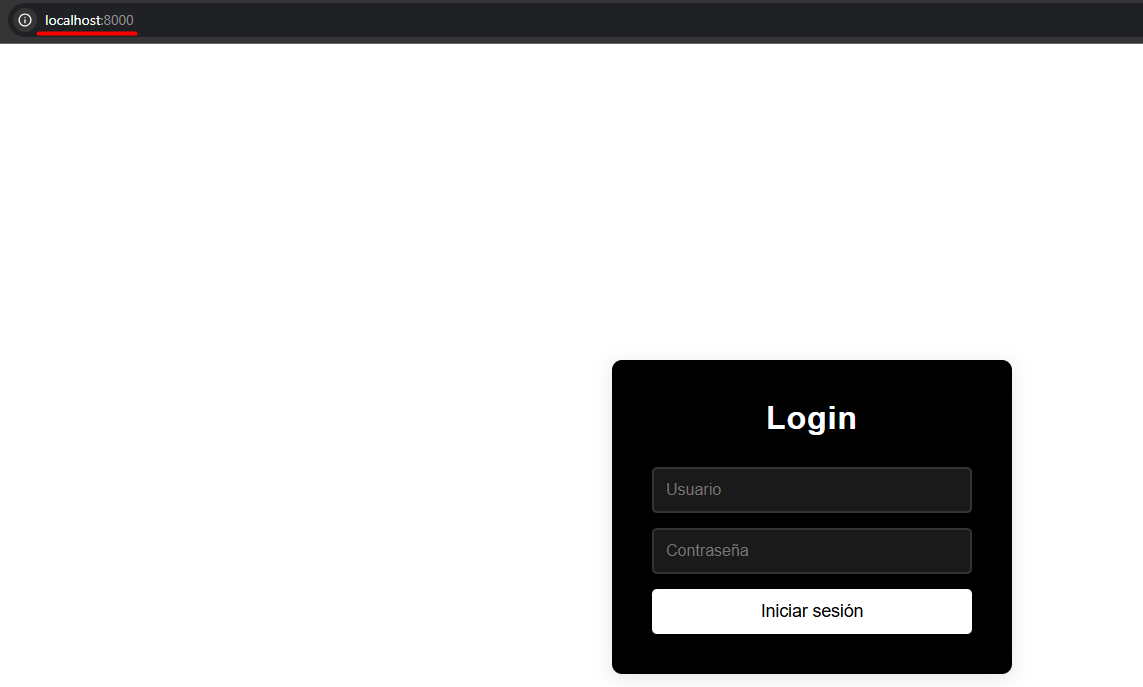
|  |
| --- |
| **npm start** |

Se puede dar el caso en el que nos dé un error al iniciar la aplicación debido a que el puerto está en uso. En este caso el puerto que se usará será el 8000, tal y como se puede ver a continuación:   
  
Si diese error se cambia el valor de esta variable y de esta manera se usará otro puerto.

# Pruebas de vulnerabilidad

## SQL Injection

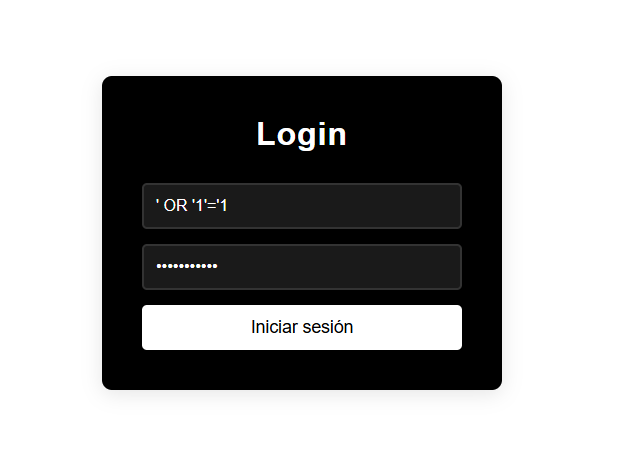
Accedemos al sitio web, como se puede observar entramos a esta insertando en el buscador “localhost:[puerto] en este caso 8000].

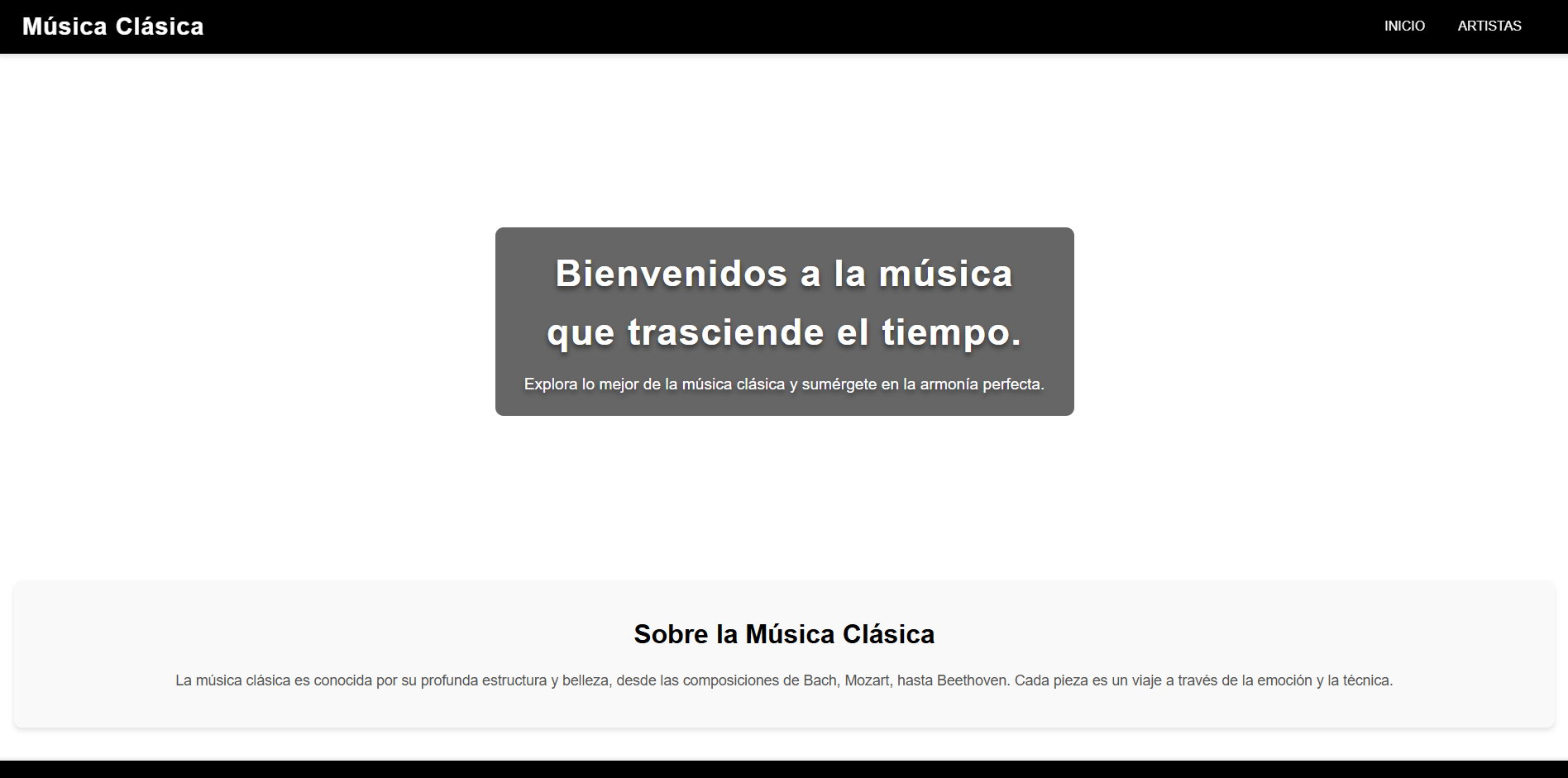


Si ponemos tanto en el nombre de usuario como en contraseña lo siguiente:

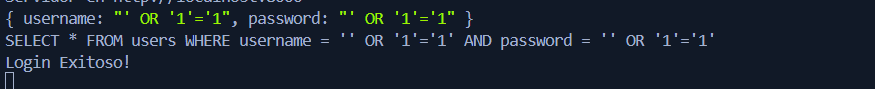
|  |
| --- |
| **' OR '1'='1** |

Veremos como podremos acceder al sitio:



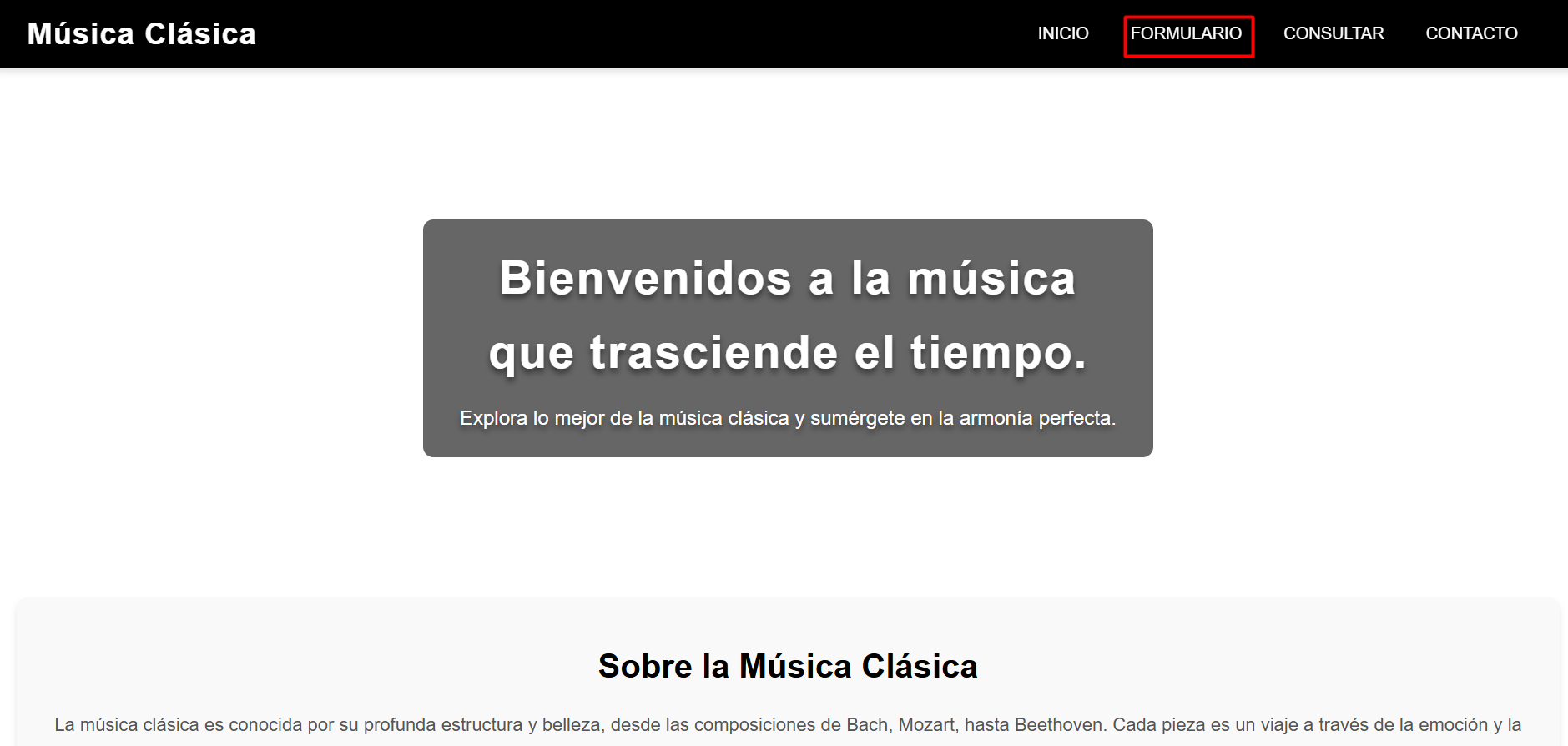


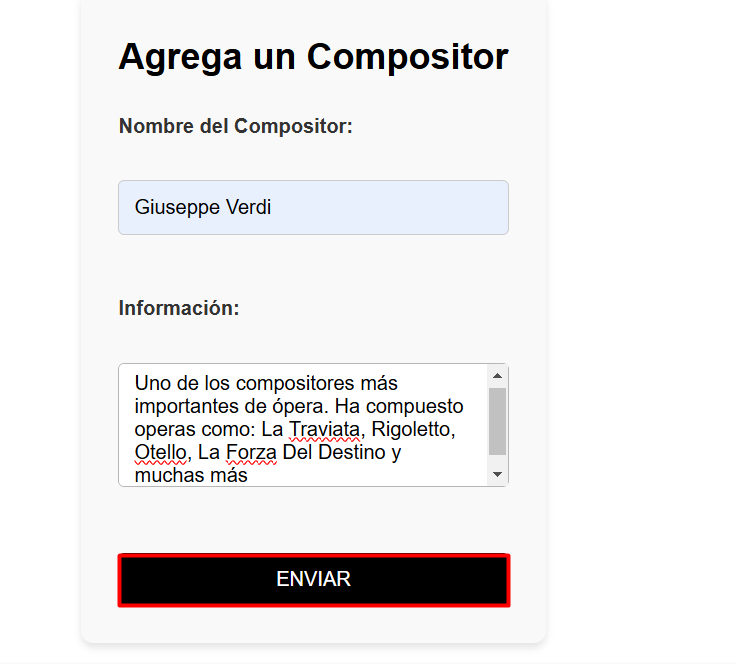
Si nos vamos al IDE podemos ver con que usuario y contraseña se ha accedido como a su vez la query que se ha usado:



Con esto queda claro que esta página es vulnerable a un ataque de **SQL Injection**.

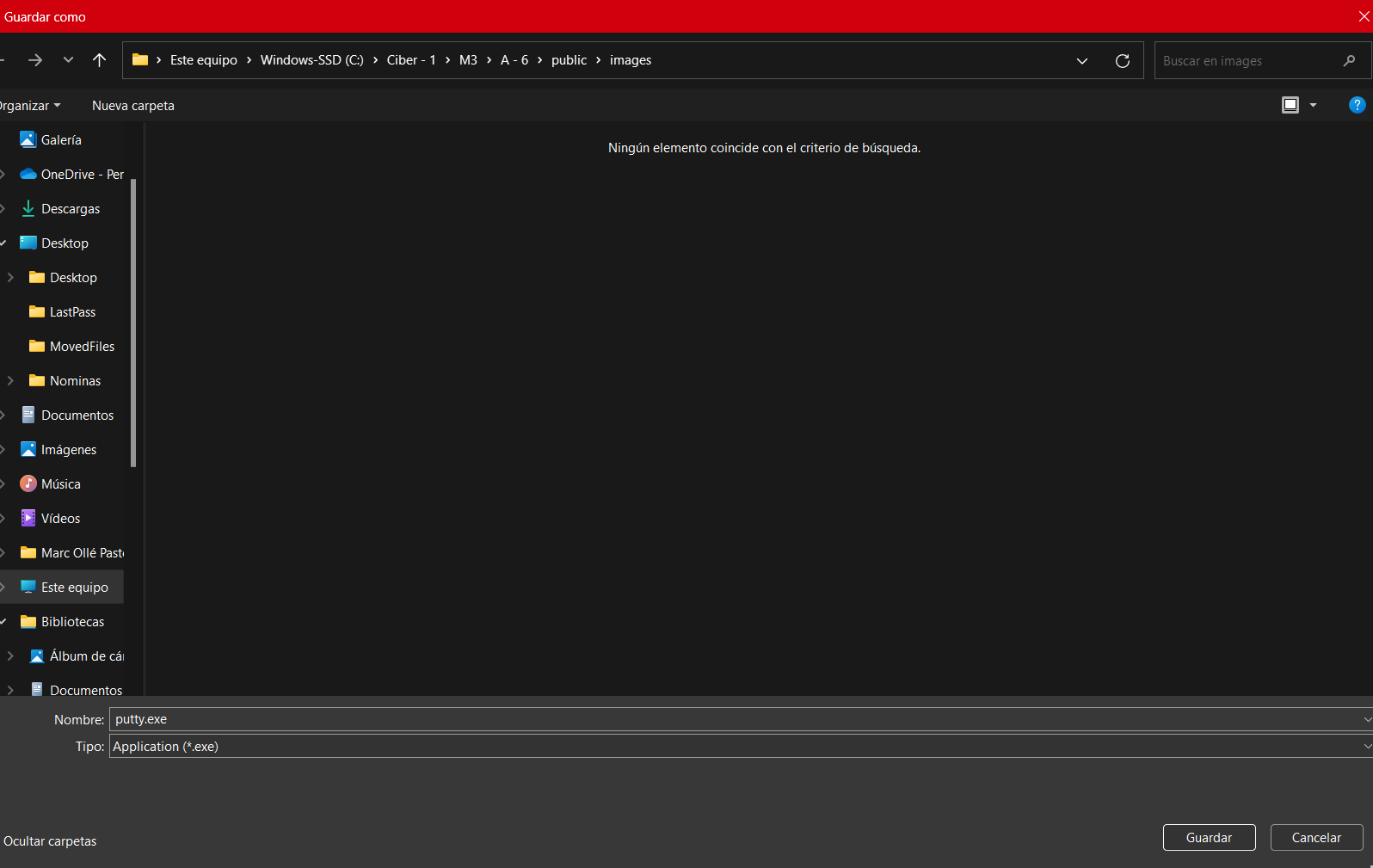
## XSS Injection

Como bien lo hemos visto en clase, los ataques XSS son un tipo de inyección en el que se inyectan secuencias maliciosas en sitios web. Estos ataques ocurren cuando un atacante utiliza una aplicación web para enviar código malicioso, generalmente en forma de una secuencia de comandos del lado del navegador a un usuario. En este caso vamos a realizar esta inyección desde el backend. La página de ClassicalFM permite introducir el conocimiento que dispone los usuarios sobre los diferentes compositores. Dentro del nav observamos como se disponen de varios enlaces a los cuales el usuario se puede dirigir. En este caso si nos dirigimos a la entrada formulario:   


Se nos abrirá una página la cual contiene un formulario. El usuario aquí pondrá el nombre del compositor y a continuación la información que disponga de este. Mostramos un ejemplo:   


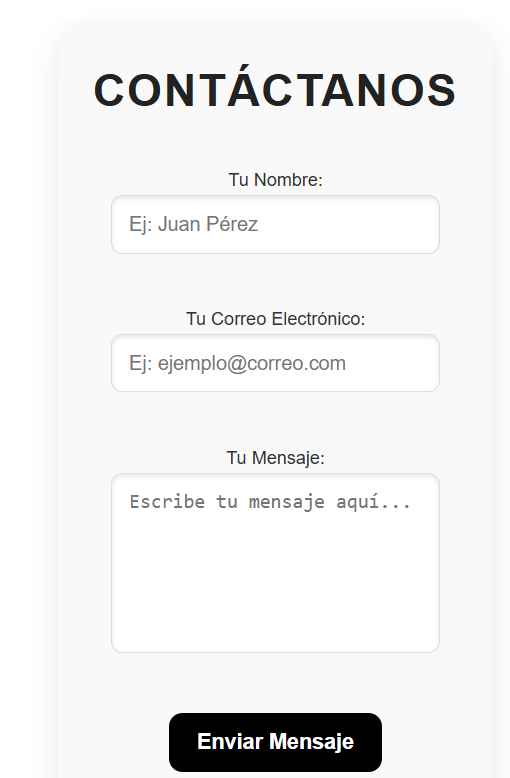
Al clicar al botón de enviar se añadirá a la base de datos los datos que ha introducido el usuario en el formulario y aparecerá la siguiente página, conforme se ha agregado correctamente el registro.

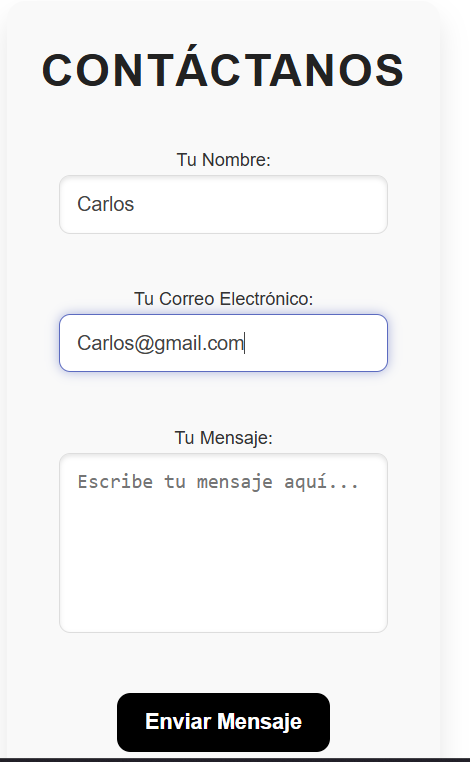


¿Pero qué sucede? Si clicamos el botón veremos que se nos descargará un fichero  
  
  
En este caso es el ejecutable del Putty pero esto podría ser cualquier fichero. Mostramos a continuación el código:   


Como se puede observar el botón tiene un Evento de escucha. Esto es como un trigger, al dar clic al botón se realizará una descarga de un fichero, concretamente el ejecutable del Putty, pero se puede hacer con cualquier fichero. El propósito es mostrar como al hacer clic se pueden ejecutar cosas. Dicho botón aparentemente hará que se vuelva a la página principal pero también hace una descarga del fichero (en este caso no, pero en un caso real sería un fichero malicioso), a su vez, como se puede apreciar hay una función de Timeout el cual hace que en 9 segundos se redirija a la página inicial por lo que dará la impresión que el botón ha hecho lo que debería hacer.

En futuras ocasiones podemos hacer que la página la cual apunte para descargar el fichero sea un servidor apache que contenga algún archivo malicioso y allí realizar la descarga de este.

A su vez mostramos otro ejemplo de XSS, disponemos de una página de contacto la cual contiene el siguiente formulario:   


Aparentemente todo está bien, pero el campo del mensaje es vulnerable a un ataque XSS. Si rellenamos los dos primeros campos no ocurrirá nada:

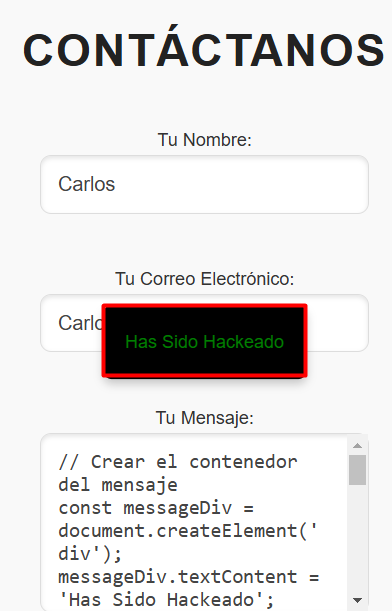
Pero la problemática viene en el campo del mensaje si introducimos el siguiente código:

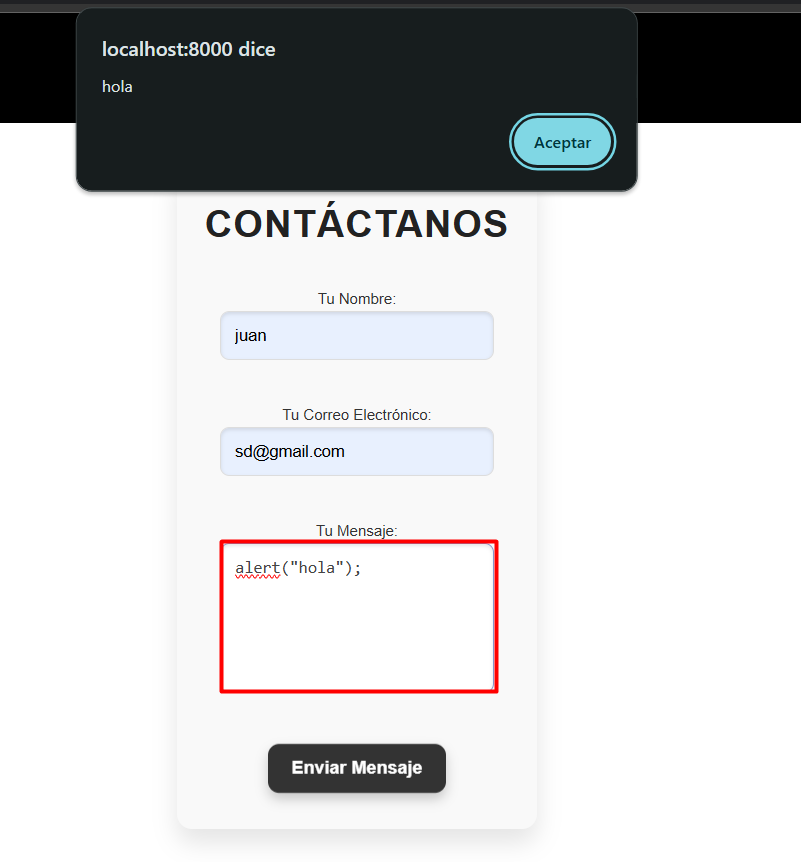
|  |
| --- |
| const messageDiv = document.createElement('div');  messageDiv.textContent = 'Has Sido Hackeado';  Object.assign(messageDiv.style, {      position: 'fixed',      top: '50%',      left: '50%',      transform: 'translate(-50%, -50%)',      padding: '20px',      backgroundColor: 'rgb(0, 0, 0)',      color: 'green',      borderRadius: '8px',      textAlign: 'center',      boxShadow: '0 4px 8px rgba(0, 0, 0, 0.2)',      fontFamily: 'Arial, sans-serif',      fontSize: '16px',  });  document.body.appendChild(messageDiv); |

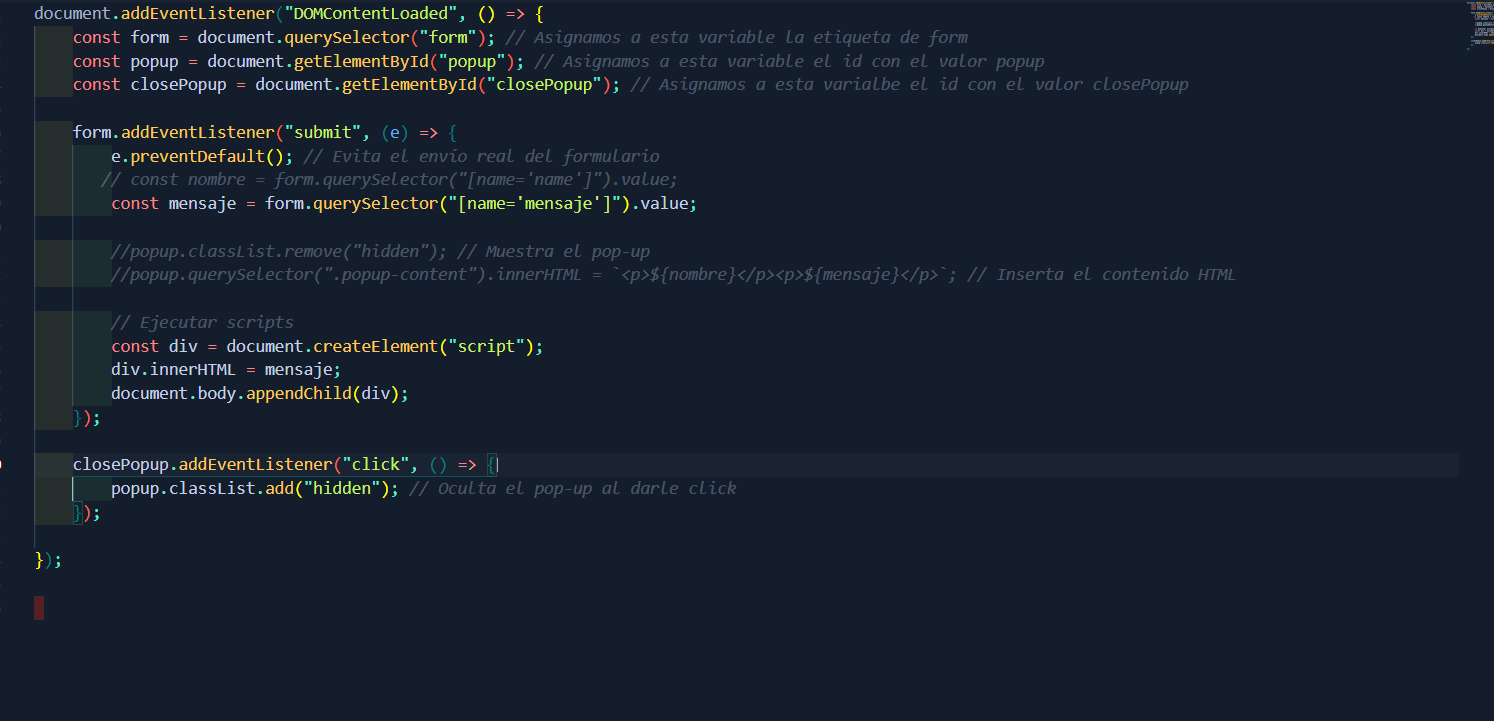
Lo introducimos y clicamos al botón de enviar mensaje:



Veremos como nos aparece un pequeño pop-up:



Todo y que no es super XSS, esto se podría considerar un ataque de Cross Site Scripting debido a que el código JS ha sido introducido desde el frontend. Podríamos hacer lo mismo pero con un simple alert.   
  


Y podemos ver como aparece el alert. El código el cual permite esto es el siguiente:   
  
Con este código permitimos la inyección XSS en la página.

## HTML Injection

Disponemos de una página en la aplicación la cual hace una búsqueda filtrada. El usuario introduce el nombre del compositor y al darle en el botón de consultar se efectuará una consulta SQL. Pero no nos adelantemos, mostramos primeramente el formulario.

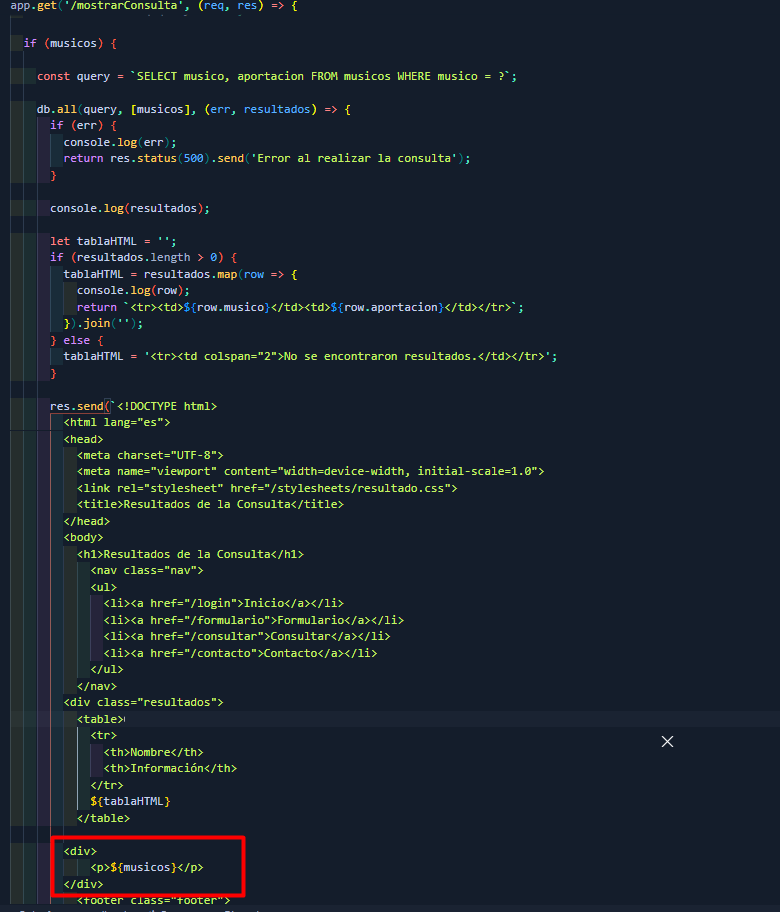


Si introducimos el nombre de un compositor veremos como se hará una búsqueda en la base de datos. Lo que sucederá será que se hará un select con una condición de where. El valor del where será el valor que hemos introducido en el formulario. Por ejemplo, si hacemos la búsqueda del compositor Giuseppe Verdi y clicamos al botón de consultar:



Veremos como muestra las aportaciones que ha puesto el usuario sobre este compositor.

  
Como se puede ver en la imagen, se muestra la info que se ha aportado sobre el compositor, pero podemos ver como en rojo aparece el nombre del compositor. Pero observemos con detenimiento, allí no debería aparecer el nombre ya que solamente ha habido 3 aportaciones sobre este y además no está dentro de la tabla. ¿Y por qué aparece? Pues bien, esto se debe a que hemos hecho un HTML Injection. Fijémonos en el código, como se observa hay un div que dentro contiene el valor que introduce el usuario en el formulario de consulta dentro de un párrafo.



HTML Injection

**Data d’entrega i format:**

Haureu d’entregar un .zip amb el codi del projecte i el PDF amb la memòria tècnica demanada al pou d’entrega de moodle abans del 14 de gener a les 23.55

**Avaluació:**

Rúbrica:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Items | 10-9 | 8-7 | 5 | 3-4 | 0-1 |
| **Introduccio (memoria)**  **(10%)** | S’ajusta el que es demana | - | - | - | no s’aporta introducció ni explicació |
| **Guia del desplegament de l’aplicació web (20%)** | Es descriu amb molt detall i precisió els passos i s’aporten imatges de qualitat | Es descriu correctament el procés | Es descriu molt breument el procés o si hi ha errors s’han documentat | L’alumne no documenta els errors i no aconsegueix una app funcional | Altres |
| **Proves de vulnerabilitat amb els formularis (20%)** | Es descriu amb molt detall i precisió els passos i s’aporten imatges de qualitat | Es descriu amb molt detall i precisió els passos i s’aporten imatges de qualitat | Es descriu molt breument el procés o si hi ha errors s’han documentat | L’alumne no documenta els errors i no aconsegueix una app funcional | Altres |
| **Implementacio de les mesures i exemples (30%)** | Es descriu amb molt detall i precisió els passos i s’aporten imatges de qualitat | Es descriu amb molt detall i precisió els passos i s’aporten imatges de qualitat | Es descriu molt breument el procés o si hi ha errors s’han documentat | L’alumne no documenta els errors i no aconsegueix una app funcional | Altres |
| **Funcionalitat del codi**  **(20%)** | el codi és funcional | - | el codi és mínimament funcional (arrenca però amb errors) | L’alumne no documenta els errors i no aconsegueix una app funcional | altres |