



NOVA

MELODY

Pablo Vigo Jiménez

Índice:

Descripción:	3
Objetivo General:	3
Objetivos Específicos:	4
Tecnologías Utilizadas:	5
Frontend:	5
HTML Y CSS:.....	5
Bootstrap:.....	5
JavaScript Nativo (ES6+):.....	6
Swiper.js:.....	6
Backend:	7
PHP:.....	7
MySQL:.....	7
Apache:.....	8
Conclusión:.....	9
Diseño de la aplicación:	10
Configuración del servidor AWS (EC2):	11
Crear una nueva instancia:.....	11
Amazon Machine Image (AMI):.....	11
Tipo de instancia:.....	12
Crear llave de acceso:.....	12
Ajustes de red:.....	13
Instalación del servidor web (APACHE):	14
Actualización de sistema y paquetes:.....	15
Instalación de Apache:.....	15
Instalación de PHP:.....	16
Iniciar y habilitar Apache:.....	16
Instalación de módulos PHP necesarios:.....	17
Verificar que Apache funcione:.....	18
Instalar la página web:	19
IP Elástica:.....	20
Crear un dominio (DUCKDNS):.....	21
Instalar certificado SSL (CERTBOT):.....	22
Configurar la base de datos (AURORA & RDS):	26
Conexión a la base de datos (HEIDISQL):.....	27
Arquitectura del Sistema:	29
¿Dónde está ubicada?.....	29
¿Cómo se llama desde el cliente?.....	30
¿Qué hace exactamente?.....	31
¿Qué archivos o partes del sistema involucra?.....	32
¿Qué datos necesita y cómo se procesan?.....	32

PROYECTO 2º DAW - CURSO: 24 / 25

Fragmentos de Código:	33
Conexión a Base de Datos (PHP):.....	33
Control de Roles y Sesiones (PHP):.....	34
Inicio de sesión (Fetch API):.....	34
Resolución de problemas:	35
Problema 1:.....	35
Solución aplicada 1:.....	35
Problema 2:.....	36
Solución aplicada 2:.....	36
Problema 3:.....	37
Solución aplicada 3:.....	37
Pliego de Condiciones:	38
Requisitos funcionales:.....	38
Requisitos técnicos:.....	38
URL Aplicación desplegada:	39
Bitácora de tareas realizadas:	39
Participación de usuarios:	40
Aportaciones positivas:.....	40
Aportaciones negativas:.....	40
Conclusiones y planes a futuro:	41
Mejorar diseño y adaptación a móviles y tablets:.....	41
Añadir más contenido para mejorar mi web:.....	42
Ofrecer opciones de personalización para los usuarios:.....	43
Mejorar la interfaz de reproducir música:.....	44
Video de demostración:	45
Bibliografía utilizada:	46

Descripción:

La idea que tengo con este **proyecto** es montar una **página web de música urbana**, al **estilo** de **Spotify**, pero hecha por mí desde cero, sobre todo he tenido mucha referencia a la plataforma de Spotify. Lo que quiero es que cualquier usuario que entre a mi web, le sea fácil e intuitivo el encontrar canciones, artistas o álbumes sin perderse, y escuchar música.

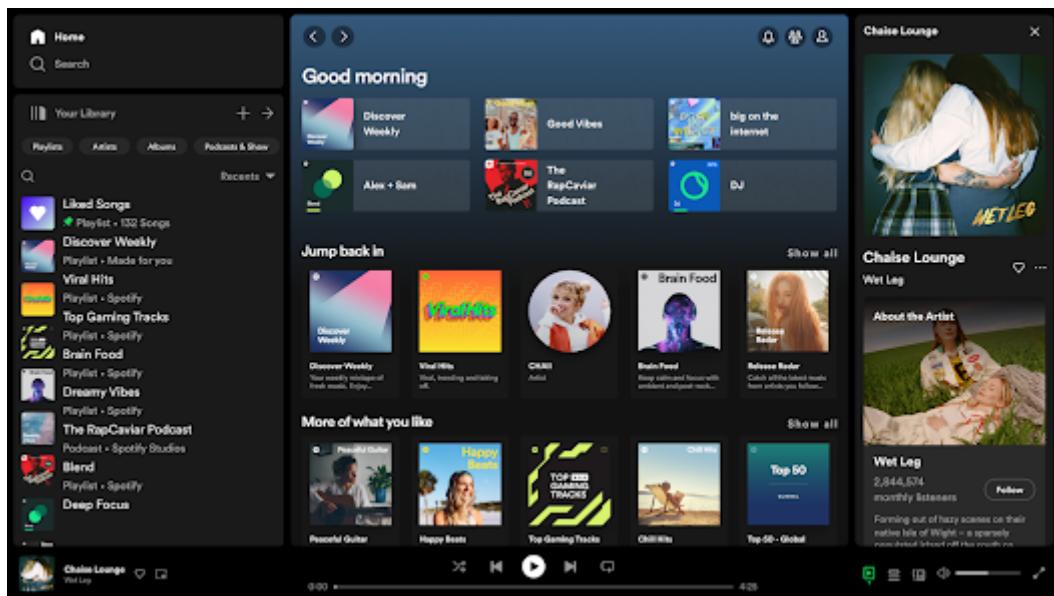
Quiero que se vea bien, tanto en un ordenador como en el móvil, que todo esté responsive y bien organizada en todos los dispositivos.

También tengo pensado **incluir** una **parte** solo para **administradores**. Desde ahí se **podrá gestionar el contenido**: subir nuevas canciones, modificar o eliminar álbumes o artistas, y tener todo bien organizado. Esto lo veo importante para que la página se mantenga **actualizada**.

Estoy haciendo este proyecto no solo como parte del aprendizaje, sino también porque me apetece crear algo propio. Algo que funcione bien, que de gusto usarlo, que sea práctico y al mismo tiempo sencillo. La idea es que tanto la gente que la use como quien la administre, se sientan cómodos con mi página.

Objetivo General:

Desarrollar una **aplicación web interactiva** de música urbana que permita a los usuarios escuchar canciones, visualizar álbumes, ver su perfil y acceder a contenido multimedia, como **inspiración** use plataformas como **Spotify**.



Objetivos Específicos:

Diseñar una interfaz moderna y adaptable:

Quiero que la web tenga un diseño muy parecido al de Spotify y quiero que se vea bien tanto en el ordenador como en el móvil, y que sea intuitiva.

Implementar una barra de búsqueda funcional:

Con la barra de búsqueda, mi principal objetivo era que se pueda buscar tanto canciones, álbumes y artistas de una manera rápida y se pueda acceder a ellos de una manera rápida y sencilla.

Conectar la web con una base de datos:

Toda la información (usuarios, canciones, álbumes, artistas...) estará almacenada de forma ordenada en una base de datos, en mi caso voy a usar MYSQL. Así todo cargará rápido y no habrá errores o datos mal gestionados.

Usar AJAX para que todo sea más dinámico:

Con AJAX puedo actualizar partes de la página sin necesidad de recargarla entera. Esto hace que todo sea más rápido y más cómodo para el usuario.

Gestionar sesiones con perfiles personalizados:

Los usuarios podrán registrarse, iniciar sesión y tener su propio perfil dentro de la plataforma. Si alguien entra como administrador, podrá controlar el contenido de la web fácilmente.

Incorporar navegación tipo SPA (Single Page Application):

Quiero que la navegación entre secciones sea rápida, sin tener que recargar toda la página cada vez. Solo se cargan los componentes necesarios, lo que mejora bastante el rendimiento.

Tecnologías Utilizadas:

Frontend:

HTML Y CSS:



Características implementadas:

- Elementos multimedia (audio, imágenes).
- CSS Grid y Flexbox para responsive.
- Animaciones CSS para transiciones suaves

Bootstrap:



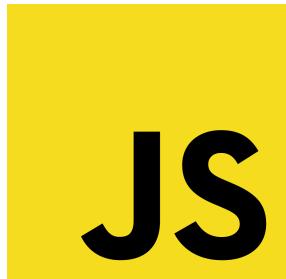
Características implementadas:

- Desarrollo rápido de interfaces responsive
- Sistema de grid para diferentes dispositivos
- Fácil de utilizar y un diseño moderno

Componentes utilizados:

- Para la cabecera de la página
- Cards para mostrar álbumes y canciones
- Sistema de grid para organización del contenido

JavaScript Nativo (ES6+):



Control total sobre el rendimiento

- Menor peso de la aplicación
- No dependencia de librerías externas pesadas
- Aprendizaje de los fundamentos del lenguaje

Funcionalidades implementadas:

- Gestión del DOM dinámico
- Validación en tiempo real de formularios
- Manejo de eventos de usuario

Swiper.js:



Características implementadas:

- Rendimiento superior comparado con Slick Slider u Owl Carousel
- Optimización móvil nativa (admite mover con gestos)
- Peso ligero

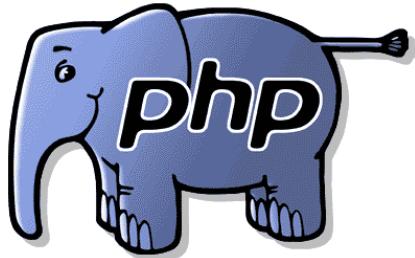
Implementación específica:

- Carrusel principal para álbumes destacados
- Navegación touch-friendly para dispositivos móviles
- Efectos de transición suaves

PROYECTO 2º DAW - CURSO: 24 / 25

Backend:

PHP:



- Para montar mi servidor con Apache, se adapta perfectamente
- Facilidad para el manejo de sesiones de usuario y gestión de roles
- Soporte nativo para subida de archivos multimedia
- Mucha documentación disponible y videos explicativos

Implementación específica:

- Gestión de sesiones para diferenciación admin/usuario
- Validación de formularios de registro y login
- Manejo seguro de archivos multimedia (validación de tipos, tamaños)

MySQL:



- Base de datos relacional ideal para gestionar relaciones entre usuarios, canciones, álbumes y artistas
- Integración perfecta con PHP mediante PDO
- Compatibilidad total con AWS RDS Aurora

Diseño específico:

- Tablas normalizadas para usuarios, canciones, álbumes, artistas
- Índices optimizados para búsquedas rápidas
- Constraints para mantener integridad referencial

PROYECTO 2º DAW - CURSO: 24 / 25

Apache:



- Un servidor web que no falle fácilmente.
- Configurar sitios seguros con certificados SSL de forma sencilla.
- Funciona muy bien para manejar archivos como música, imágenes y videos.
- Tiene extras que ayudan a que la página cargue más rápido y use mejor los recursos

Conclusión:

Elegí estas tecnologías para usarlas en mi proyecto por varias razones, entre ellas:

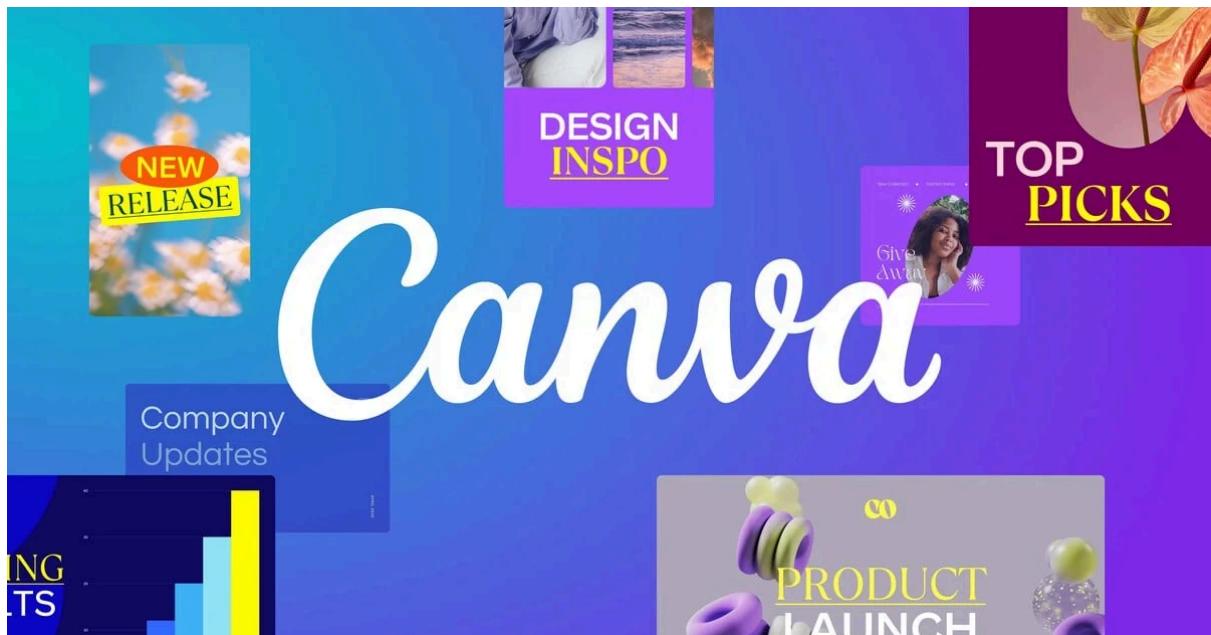
- Mejor rendimiento para que funcione más rápido mi página
- Mantenibilidad a largo plazo
- Mejorar mi experiencia con estas tecnologías



Diseño de la aplicación:

Para este apartado me apoyé en **Canva**, ya que la **plataforma** me permite **exportar** el diseño en varios formatos, como **PDF**, lo cual me vino genial. Usé Canva para **plasmar** de forma visual una **idea general** de cómo fue tomando forma mi proyecto desde el principio.

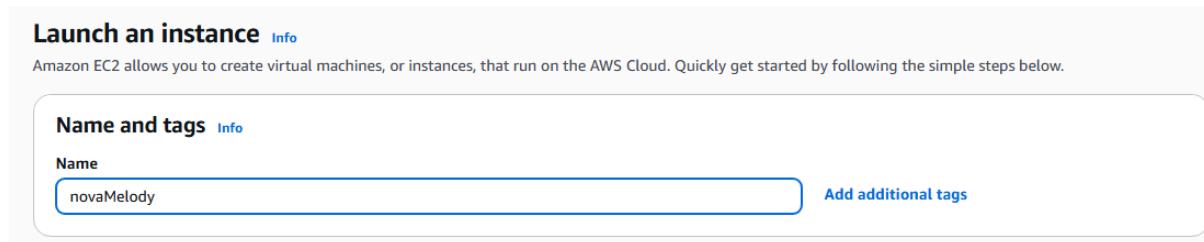
Desde este [enlace](#) se puede encontrar el diseño de la aplicación.



Configuración del servidor AWS (EC2):

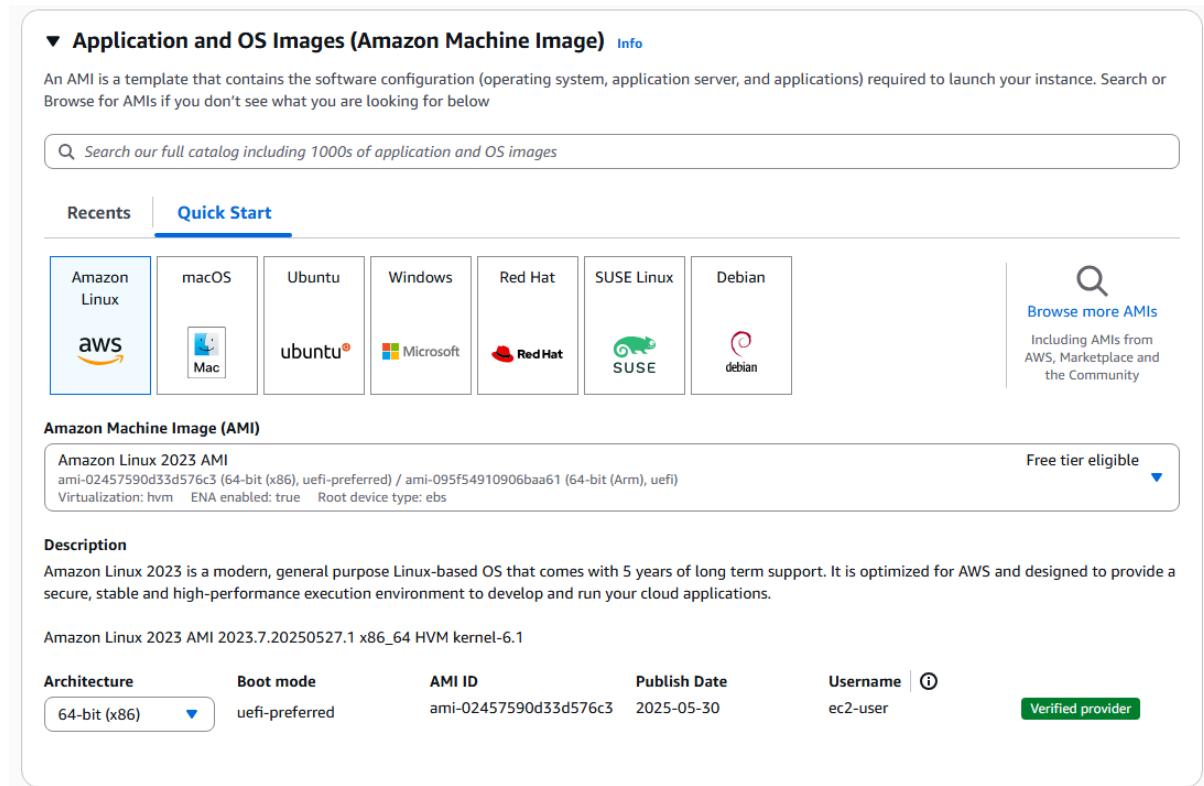
Lo primero que vamos a hacer es **dirigirnos a la barra de búsqueda** que aparece arriba a la izquierda de la página y **buscaremos EC2, creamos una instancia nueva**, ahora procederemos a seguir los pasos de la captura de pantalla a continuación:

Crear una nueva instancia:



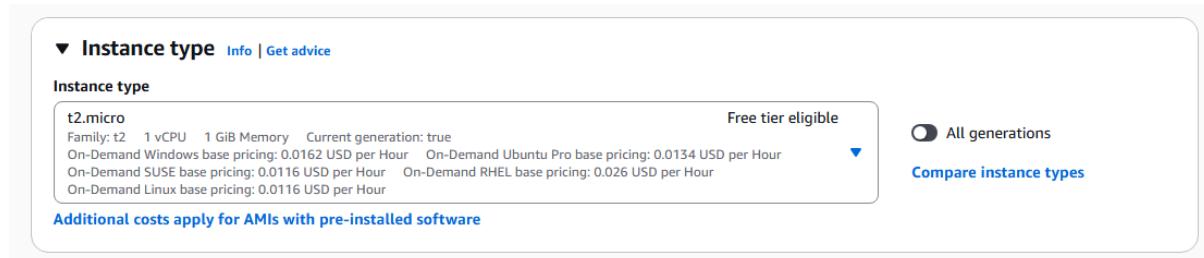
Amazon Machine Image (AMI):

Primero elegimos el **nombre de nuestra instancia**, en mi caso '**novaMelody**', el siguiente paso será dejarlo por defecto, ya que este es el servidor que se va a utilizar, en mi caso **Amazon Linux 2023 AMI**, como se observa en la captura de pantalla:



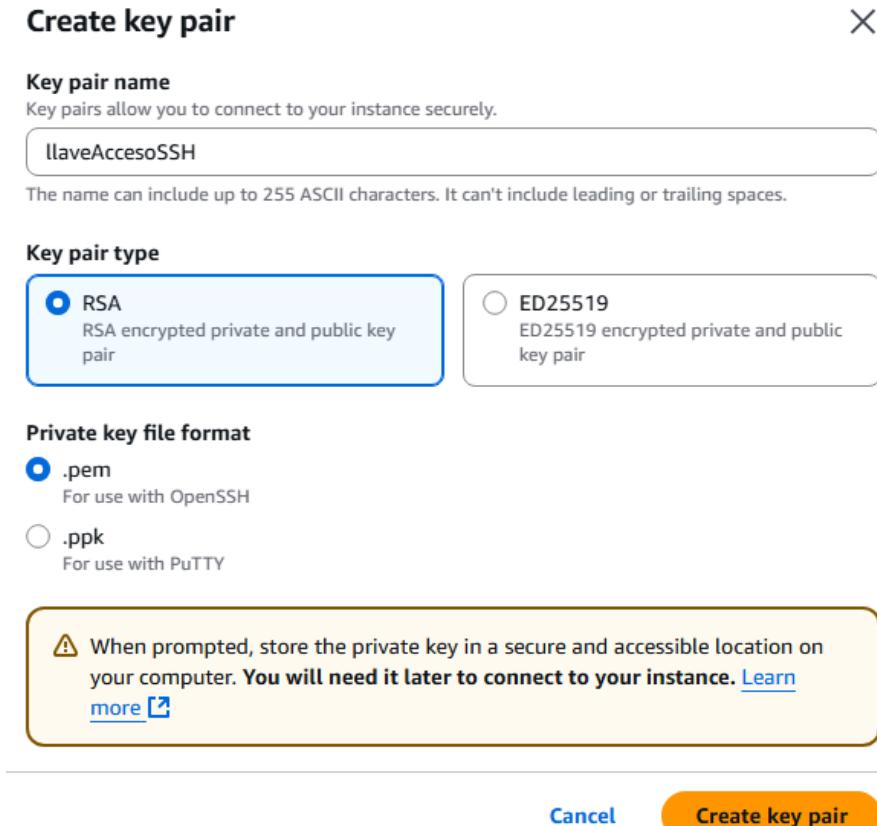
Tipo de instancia:

El **siguiente paso** es **elegir** el **tipo de instancia**, que lo dejaremos por defecto, ya que es el gratuito y con los requisitos de **memoria** y **CPU** que nos aportan nos sirven perfectamente.



Crear llave de acceso:

Ahora procederemos a **crear** una **llave de acceso** para poder **conectarnos** de manera remota al servidor **mediante SSH**, se adjuntan capturas de pantalla de la configuración a seguir:



Ajustes de red:

El siguiente apartado es **ajustes de la red**, bien, aquí debemos de ajustar estas configuraciones y **muy importante**, debemos de **tener activado estas tres casillas** para que **automáticamente se creen los puertos de cada protocolo y habilitar también la conexión desde cualquier tipo de IP** (habilitarlo y hacerlo público para todo el mundo).

The screenshot shows the 'Network settings' configuration page for an AWS instance. At the top, there are sections for 'Network' (vpc-0b2553cb9d78ad644), 'Subnet' (No preference), and 'Auto-assign public IP' (Enable). Below these, a 'Firewall (security groups)' section is shown with a note about security groups controlling traffic to the instance. Two options are available: 'Create security group' (selected) and 'Select existing security group'. A note below states: 'We'll create a new security group called 'launch-wizard-5' with the following rules:'. Under this, three checkboxes are checked: 'Allow SSH traffic from Anywhere (0.0.0.0/0)', 'Allow HTTPS traffic from the internet (To set up an endpoint, for example when creating a web server)', and 'Allow HTTP traffic from the internet (To set up an endpoint, for example when creating a web server)'. A warning message at the bottom right says: '⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' with a close button 'X'.

Instalación del servidor web (APACHE):

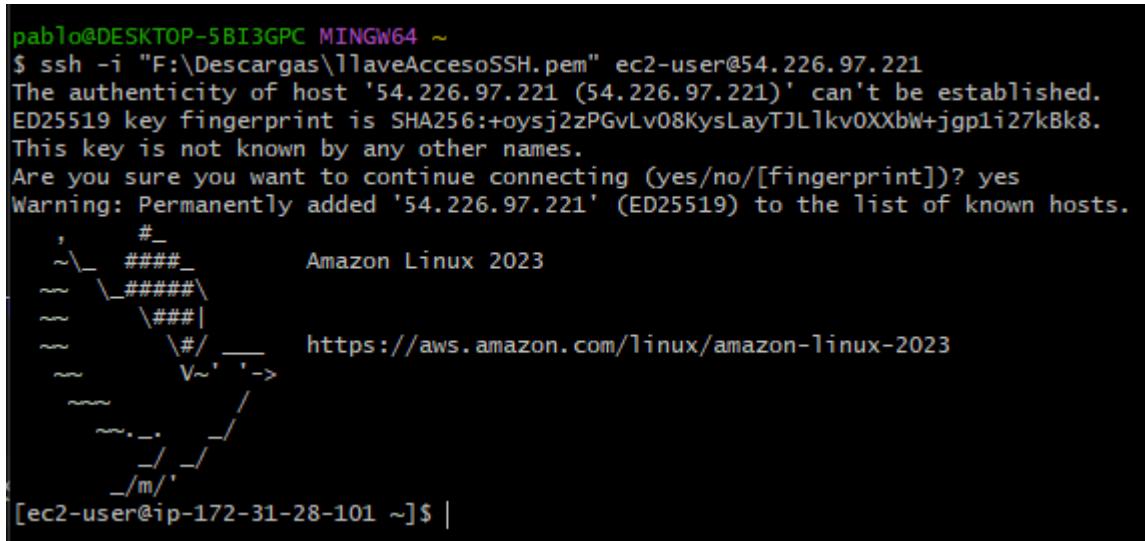
Lo primero que haremos será conectarnos a través de SSH utilizando la clave de acceso vista en el [anterior apartado](#), para mi caso utilizaré este [programa](#).

Una vez tenemos todo ya instalado, procederemos a conectarnos a través de SSH finalmente, adjunto captura de pantalla a seguir los pasos.



```
pablo@DESKTOP-5BI3GPC MINGW64 ~
$ ssh -i "F:\Descargas\llaveAccesoSSH.pem" ec2-user@54.226.97.221
```

Si nos aparece la siguiente pantalla, eso significa que lo hicimos correctamente, así que proseguiremos con los siguientes pasos.



```
pablo@DESKTOP-5BI3GPC MINGW64 ~
$ ssh -i "F:\Descargas\llaveAccesoSSH.pem" ec2-user@54.226.97.221
The authenticity of host '54.226.97.221 (54.226.97.221)' can't be established.
ED25519 key fingerprint is SHA256:+oysj2zPGvLv08KysLayTJL1kv0XXbW+jgp1i27k8k8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.226.97.221' (ED25519) to the list of known hosts.

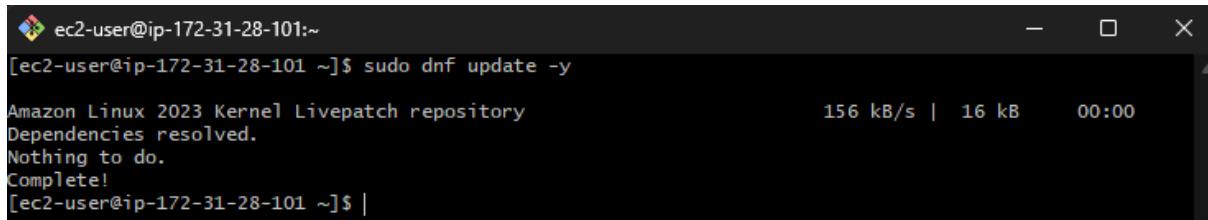
,      #
~\_ #####
~~ \####\
~~  \###|
~~   \#/ ,__->
~~    V~, .-'-
~~     / \
~~    / /
~~   / /
~~  / /
[ec2-user@ip-172-31-28-101 ~]$ |
```

Actualización de sistema y paquetes:

El siguiente paso será el actualizar sistema y los paquetes, esto se hace para evitar futuros problemas de versiones incompatibles.

Comando:

- sudo dnf update -y



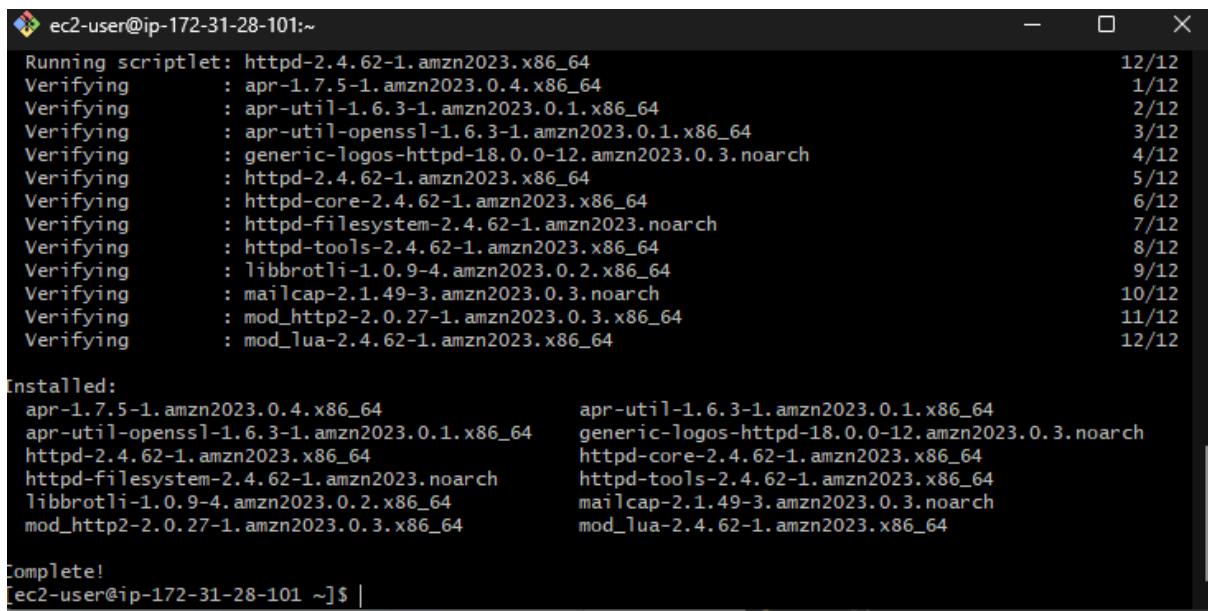
```
ec2-user@ip-172-31-28-101:~ [ec2-user@ip-172-31-28-101 ~]$ sudo dnf update -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-28-101 ~]$ |
```

Instalación de Apache:

El siguiente paso sería el instalar el servidor web Apache

Comando:

- sudo dnf install -y httpd



```
ec2-user@ip-172-31-28-101:~ [ec2-user@ip-172-31-28-101 ~]$ sudo dnf install -y httpd
Running scriptlet: httpd-2.4.62-1.amzn2023.x86_64
Verifying : apr-1.7.5-1.amzn2023.0.4.x86_64 12/12
Verifying : apr-util-1.6.3-1.amzn2023.0.1.x86_64 1/12
Verifying : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 2/12
Verifying : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 3/12
Verifying : httpd-2.4.62-1.amzn2023.x86_64 4/12
Verifying : httpd-core-2.4.62-1.amzn2023.x86_64 5/12
Verifying : httpd-filesystem-2.4.62-1.amzn2023.noarch 6/12
Verifying : httpd-tools-2.4.62-1.amzn2023.x86_64 7/12
Verifying : libbrotli-1.0.9-4.amzn2023.0.2.x86_64 8/12
Verifying : mailcap-2.1.49-3.amzn2023.0.3.noarch 9/12
Verifying : mod_http2-2.0.27-1.amzn2023.0.3.x86_64 10/12
Verifying : mod_lua-2.4.62-1.amzn2023.x86_64 11/12
Verifying : mod_lua-2.4.62-1.amzn2023.x86_64 12/12
Installed:
  apr-1.7.5-1.amzn2023.0.4.x86_64      apr-util-1.6.3-1.amzn2023.0.1.x86_64
  apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  httpd-2.4.62-1.amzn2023.x86_64        httpd-core-2.4.62-1.amzn2023.x86_64
  httpd-filesystem-2.4.62-1.amzn2023.noarch  httpd-tools-2.4.62-1.amzn2023.x86_64
  libbrotli-1.0.9-4.amzn2023.0.2.x86_64   mailcap-2.1.49-3.amzn2023.0.3.noarch
  mod_http2-2.0.27-1.amzn2023.0.3.x86_64  mod_lua-2.4.62-1.amzn2023.x86_64
Complete!
[ec2-user@ip-172-31-28-101 ~]$ |
```

Instalación de PHP:

El siguiente paso, será instalar PHP básico:

Comando:

- sudo dnf install -y php

```
[ec2-user@ip-172-31-28-101 ~]$ sudo dnf install -y php
Last metadata expiration check: 0:07:56 ago on Thu Jun  5 16:22:11 2025.
Dependencies resolved.

=====
Package           Architecture Version       Repository  Size
=====
Installing:
php8.4           x86_64      8.4.6-1.amzn2023.0.1   amazonlinux 17 k
Installing dependencies:
libsodium         x86_64      1.0.19-4.amzn2023
libxml           x86_64      1.1.43-1.amzn2023.0.1   amazonlinux 183 k
nginx-filesystem noarch     1:1.26.3-1.amzn2023.0.1   amazonlinux 9.6 k
php8.4-cli        x86_64      8.4.6-1.amzn2023.0.1   amazonlinux 3.5 M
php8.4-common     x86_64      8.4.6-1.amzn2023.0.1   amazonlinux 773 k
php8.4-process    x86_64      8.4.6-1.amzn2023.0.1   amazonlinux 46 k
php8.4-xml        x86_64      8.4.6-1.amzn2023.0.1   amazonlinux 908 k
Installing weak dependencies:
php8.4-fpm        x86_64      8.4.6-1.amzn2023.0.1   amazonlinux 1.9 M
php8.4-mbstring   x86_64      8.4.6-1.amzn2023.0.1   amazonlinux 534 k
php8.4-opcache    x86_64      8.4.6-1.amzn2023.0.1   amazonlinux 480 k
php8.4-pdo         x86_64      8.4.6-1.amzn2023.0.1   amazonlinux 94 k
php8.4-sodium     x86_64      8.4.6-1.amzn2023.0.1   amazonlinux 45 k
```

Iniciar y habilitar Apache:

Ahora lo que haremos será iniciar y habilitar Apache para que empiece a funcionar correctamente.

Comando:

- sudo systemctl start httpd
- sudo systemctl enable httpd

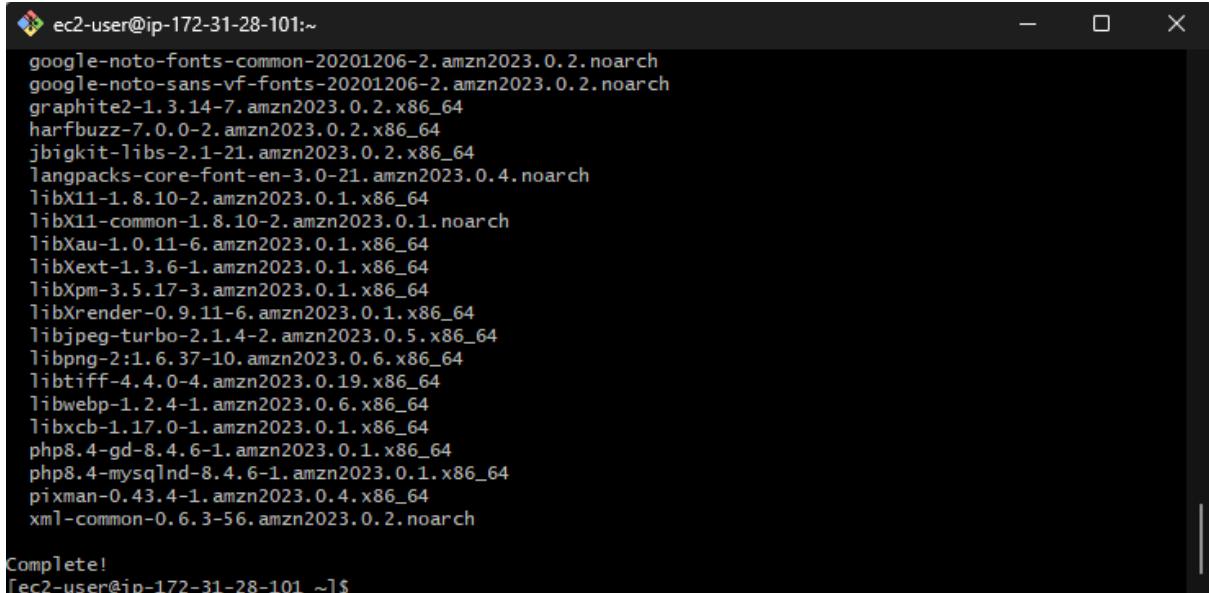
```
[ec2-user@ip-172-31-28-101 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-28-101 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-28-101 ~]$
```

Instalación de módulos PHP necesarios:

Por último instalamos los módulos más usados para bases de datos, texto, imágenes y más:

Comando:

- sudo dnf install -y php-mysqli php-pdo php-pdo_mysql php-mbstring php-xml
php-gd unzip



The screenshot shows a terminal window with the following text output:

```
ec2-user@ip-172-31-28-101:~  
google-noto-fonts-common-20201206-2.amzn2023.0.2.noarch  
google-noto-sans-vf-fonts-20201206-2.amzn2023.0.2.noarch  
graphite2-1.3.14-7.amzn2023.0.2.x86_64  
harfbuzz-7.0.0-2.amzn2023.0.2.x86_64  
jbigkit-libs-2.1-21.amzn2023.0.2.x86_64  
langpacks-core-font-en-3.0-21.amzn2023.0.4.noarch  
libX11-1.8.10-2.amzn2023.0.1.x86_64  
libX11-common-1.8.10-2.amzn2023.0.1.noarch  
libXau-1.0.11-6.amzn2023.0.1.x86_64  
libXext-1.3.6-1.amzn2023.0.1.x86_64  
libXpm-3.5.17-3.amzn2023.0.1.x86_64  
libXrender-0.9.11-6.amzn2023.0.1.x86_64  
libjpeg-turbo-2.1.4-2.amzn2023.0.5.x86_64  
libpng-2:1.6.37-10.amzn2023.0.6.x86_64  
libtiff-4.4.0-4.amzn2023.0.19.x86_64  
libwebp-1.2.4-1.amzn2023.0.6.x86_64  
libxcb-1.17.0-1.amzn2023.0.1.x86_64  
php8.4-gd-8.4.6-1.amzn2023.0.1.x86_64  
php8.4-mysqli-8.4.6-1.amzn2023.0.1.x86_64  
pixman-0.43.4-1.amzn2023.0.4.x86_64  
xml-common-0.6.3-56.amzn2023.0.2.noarch  
  
Complete!  
[ec2-user@ip-172-31-28-101 ~]$
```

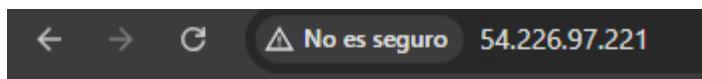
Verificar que Apache funcione:

Para ello lo que vamos a hacer es **dirigirnos** a **nuestro navegador** y **escribir** la **dirección IPv4 pública**, en mi caso es **54.226.97.221**, como se muestra en la siguiente imagen:

Public IPv4 address
 54.226.97.221 | [open address](#) 

Bien, ahora que tenemos la **dirección IPv4 pública**, vamos al navegador y copiamos y pegamos ahí mismo la **dirección IPv4 pública**.

Si hicimos todo el **procedimiento bien**, debemos de ver que **Apache funciona** y cuando ingresamos a la web pone lo siguiente:



It works!

Instalar la página web:

Para empezar a instalar la página web, lo que debemos de hacer es instalar nuestro **archivo zip** (este se localiza en nuestro GitHub) y **acceder** a esta ruta en nuestro servidor Apache '**/var/www/html/**', una vez ahí descomprimimos nuestro zip, le damos permisos a nuestra carpeta, tal y como se muestra en la imagen adjunta:

```
$ sudo chown -R www-data:www-data /var/www/html|
```

y por último, reiniciamos el servicio de Apache.

```
$ sudo systemctl restart apache2|
```

```
ec2-user@ip-172-31-28-101:/var/www/html
[ec2-user@ip-172-31-28-101 html]$ ls -la
total 36
drwxr-xr-x. 7 apache apache 102 Jun  5 16:46 .
drwxr-xr-x. 4 root   root    33 Jun  5 16:28 ..
drwxr-xr-x. 4 apache apache 16384 Mar 23 20:48 PHPMailer
drwxr-xr-x. 6 apache apache  51 Feb 23 16:51 assets
drwxr-xr-x. 2 apache apache 16384 Mar 30 17:56 controladores
drwxr-xr-x. 2 apache apache   84 Mar 26 19:22 includes
-rw xr-xr-x. 1 apache apache 1665 Jun  3 00:04 index.php
drwxr-xr-x. 2 apache apache  164 Mar 20 21:13 php
[ec2-user@ip-172-31-28-101 html]$ |
```

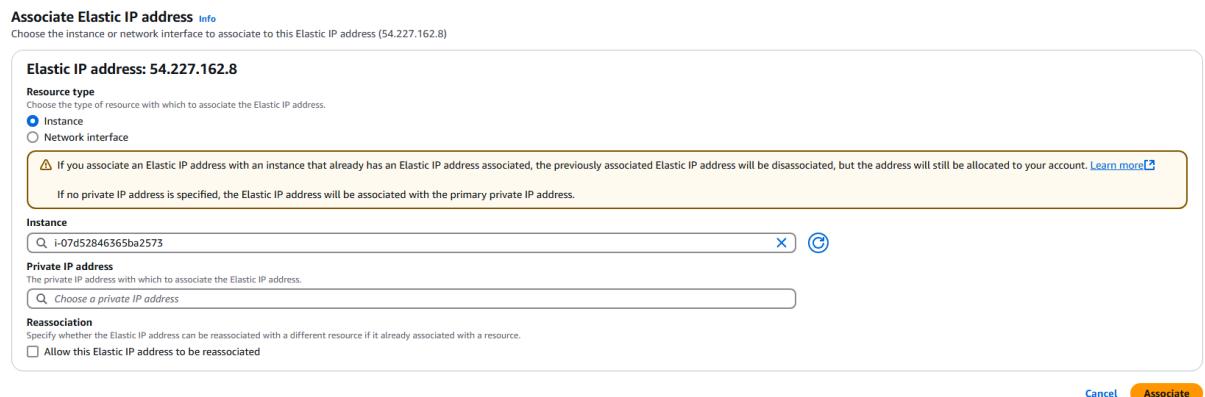
PROYECTO 2º DAW - CURSO: 24 / 25

IP Elástica:

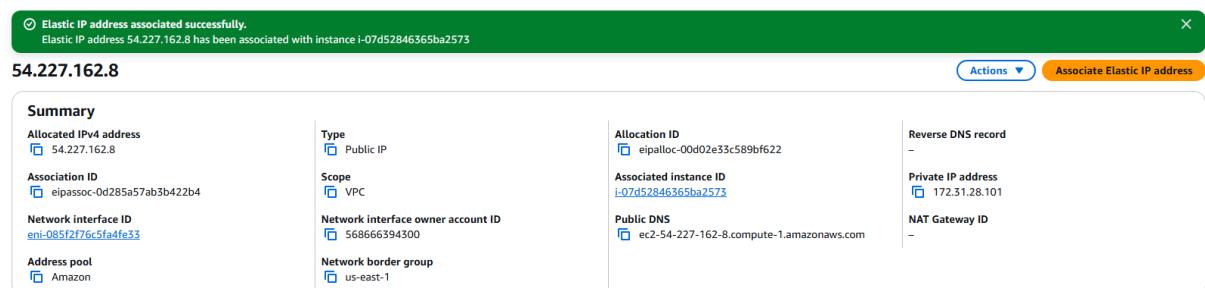
Para poder **asociar una IP elástica** a nuestra máquina virtual, primero debemos de ir a la sección de **IP elástica** y una vez abierta la sección, nos desplazamos a 'Instancias' y ahí observamos nuestra máquina que está ahora mismo conectada para que se determine la IP elástica.



Nos deberá de quedar tal que así **toda la configuración**, como en la captura adjunta.



Una vez asociada, ahora **ya tendremos una IP elástica** en nuestro servidor.



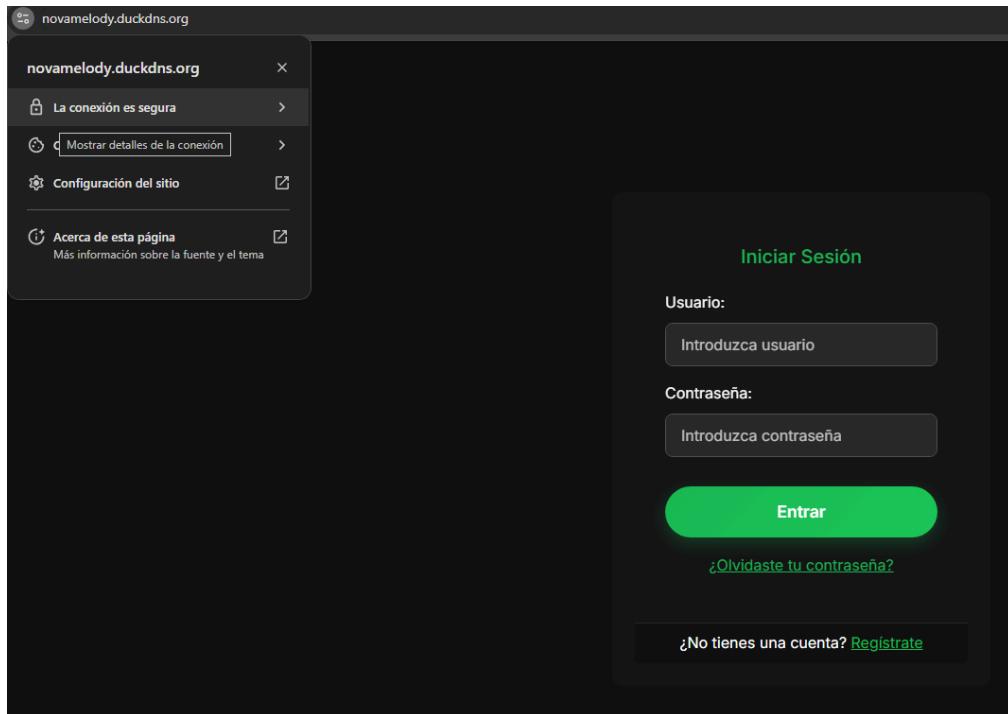
PROYECTO 2º DAW - CURSO: 24 / 25

Crear un dominio (DUCKDNS):

Para ello me voy a ayudar de la siguiente tecnología, [DuckDNS](#), para poder obtener un dominio gratuito y para más adelante poder instalar un [certificado SSL](#) al completo.

The screenshot shows the Duck DNS interface. At the top, there's a yellow rubber duck logo and account information: account espacioextra43@gmail.com, type free, token f857d491-710e-488e-a284-42635c2e7546, token generated 10 minutes ago, and created date 5 Jun 2025, 18:10:03. Below this, a green bar displays the message "success: ip address for novamelody.duckdns.org updated to 54.226.97.221". The main area is titled "domains" with a count of 15. It has tabs for "http://", "sub domain", "duckdns.org", and "add domain". A table lists domains with columns: domain, current ip, ipv6, and changed. One row for "novamelody" shows the current IP as 54.226.97.221, an empty ipv6 field, and a change timestamp of "0 seconds ago". Buttons for "update ip" and "update ipv6" are present. At the bottom, a note states: "This site is protected by reCAPTCHA and the Google [Privacy Policy](#) and [Terms of Service](#) apply."

Puedes acceder a este otro enlace, si quieras probar entrar desde la dirección IP elástica.



Instalar certificado SSL (CERTBOT):

En este caso utilizaré para instalar el certificado de seguridad [Certbot](#), ya que este te permite instalar un **certificado SSL** de una manera sencilla y óptima para la página web.

Mediante esta línea de comandos, podremos instalar este complemento para el certificado.

Comando:

- sudo dnf install certbot python3-certbot-apache -y

```
[ec2-user@ip-172-31-28-101 ~]$ sudo dnf install certbot python3-certbot-apache -y
Last metadata expiration check: 1:18:52 ago on Thu Jun  5 16:22:11 2023.
Dependencies resolved.

Package           Architecture Version      Repository  Size
Installing:
certbot            noarch    2.6.0-4.amzn2023.0.1   amazonlinux 49 k
certbot-certbot-apache  noarch    2.6.0-4.amzn2023.0.1   amazonlinux 287 k
Installing dependencies:
augsas-l1he        x86_64   1.13.0-1.amzn2023.0.2   amazonlinux 408 k
fontawsome-fonts   noarch   1.14.7.0-11.amzn2023.0.2   amazonlinux 205 k
mod_ssl            x86_64   1.2.4.62-1.amzn2023   amazonlinux 112 k
python3-acme       noarch   2.0.0-1.amzn2023.0.1   amazonlinux 163 k
python3-augssas    noarch   1.1.0-10.amzn2023   amazonlinux 34 k
python3-certbot    noarch   2.6.0-4.amzn2023.0.1   amazonlinux 677 k
python3-configargparse noarch  1.7.1-amzn2023   amazonlinux 45 k
python3-icmprypt   noarch   1.13.0-6.amzn2023   amazonlinux 61 k
python3-pycparser  noarch   2.24.0-1.amzn2023   amazonlinux 90 k
python3-pyopenssl  noarch   21.0.0-1.amzn2023.0.2   amazonlinux 92 k
python3-pyrfc339   noarch   1.1-16.amzn2023   amazonlinux 19 k
ascq               x86_64   3.0.3-76.amzn2023   amazonlinux 45 k
Installing weak dependencies:
python-josepy-doc  noarch   1.13.0-6.amzn2023   amazonlinux 20 k

Transaction Summary

Install 15 Packages

Total download size: 2.2 M
Installed size: 9.5 M
Downloading Packages:
(1/15): augsas-l1he-1.13.0-1.amzn2023.0.2.x86_64.rpm          5.2 MB/s | 408 kB  00:00
(2/15): certbot-2.6.0-4.amzn2023.0.1.noarch.rpm                606 kB/s | 49 kB  00:00
(3/15): fontawsome-fonts-1.14.7.0-11.amzn2023.0.2.noarch.rpm   2.1 MB/s | 205 kB  00:00
(4/15): mod_ssl-2.4.62-1.amzn2023.x86_64.rpm                 4.1 MB/s | 112 kB  00:00
(5/15): python3-acme-2.0.0-1.amzn2023.0.1.noarch.rpm         0.43 MB/s | 15 kB  00:00
(6/15): python3-augssas-1.1.0-10.amzn2023.noarch.rpm         1.3 MB/s | 34 kB  00:00
(7/15): python3-acme-2.6.0-4.amzn2023.0.1.noarch.rpm         2.2 MB/s | 161 kB  00:00
(8/15): python3-certbot-2.6.0-4.amzn2023.0.1.noarch.rpm       8.4 MB/s | 677 kB  00:00
(9/15): python3-certbot-apache-2.6.0-4.amzn2023.0.1.noarch.rpm 4.6 MB/s | 287 kB  00:00
(10/15): python3-configargparse-1.7.1-amzn2023.noarch.rpm     1.7 MB/s | 46 kB  00:00
```

PROYECTO 2º DAW - CURSO: 24 / 25

Ahora procederemos a ejecutar el [Certbot](#), nos pedirá ingresar un email, aceptamos los términos y condiciones, después nos pedirá si queremos compartir nuestro email, en mi caso no y por último nos pedirá introducir el [dominio](#).

Comando:

- sudo certbot --apache

```
[ec2-user@ip-172-31-28-101 ~]$ sudo certbot --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): espacioextra8@gmail.com
```

```
[ec2-user@ip-172-31-28-101 ~]$ sudo certbot --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): espacioextra8@gmail.com

-----
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.5-February-24-2025.pdf. You must
agree in order to register with the ACME server. Do you agree?
-----
(Y)es/ (N)o: Y
```

```
[ec2-user@ip-172-31-28-101 ~]$ sudo certbot --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): espacioextra8@gmail.com

-----
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.5-February-24-2025.pdf. You must
agree in order to register with the ACME server. Do you agree?
-----
(Y)es/ (N)o: Y

-----
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
-----
(Y)es/ (N)o: N
```

PROYECTO 2º DAW - CURSO: 24 / 25

Procedemos a **crear un VirtualHost** para **configurar nuestro sitio web**, en mi caso, crearé la siguiente configuración para poder **instalar el SSL** tanto a nuestro dominio como a **nuestra IP elástica**, entre otras configuraciones que se le pueden proporcionar. A continuación se muestran capturas de pantalla:

```
GNU nano 8.3
/etc/httpd/conf.d/novamelody.conf
<VirtualHost *:80>
    ServerName novamelody.duckdns.org
    DocumentRoot /var/www/html
    ErrorLog /var/log/httpd/novamelody_error.log
    CustomLog /var/log/httpd/novamelody_access.log combined
</VirtualHost>
```

```
<VirtualHost *:443>
    ServerName novamelody.duckdns.org

    DocumentRoot /var/www/html

    SSLEngine on
    SSLCertificateFile /etc/letsencrypt/live/novamelody.duckdns.org/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/novamelody.duckdns.org/privkey.pem

    <Directory /var/www/html>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog /var/log/httpd/novamelody_ssl_error.log
    CustomLog /var/log/httpd/novamelody_ssl_access.log combined
</VirtualHost>
```

En el de la **IP elástica**, utilicé el certificado SSL autofirmado por un certificado generado localmente por Apache y por tanto no utiliza el [Certbot](#) visto anteriormente, sino que es un SSL autofirmado por Apache.

```
<VirtualHost *:443>
    ServerName 54.227.162.8

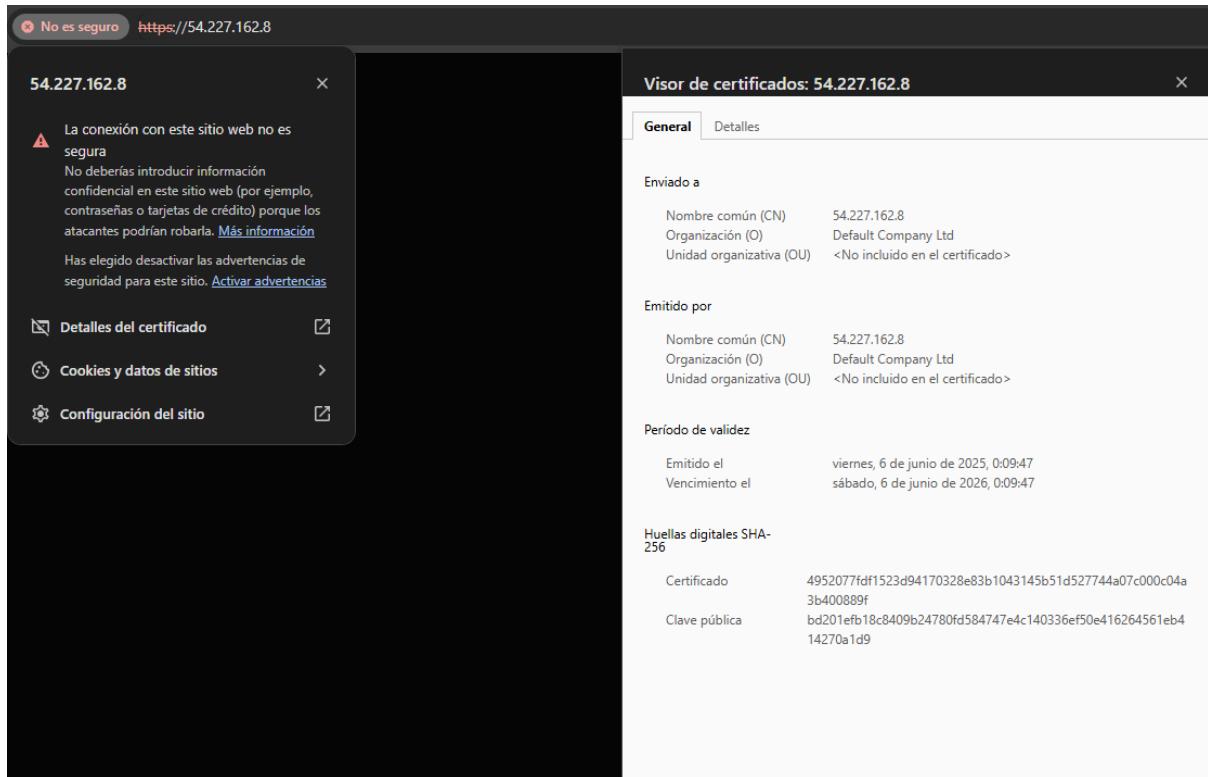
    DocumentRoot /var/www/html

    SSLEngine on
    SSLCertificateFile /etc/pki/tls/certs/localhost.crt
    SSLCertificateKeyFile /etc/pki/tls/private/localhost.key

    ErrorLog /var/log/httpd/ip_ssl_error.log
    CustomLog /var/log/httpd/ip_ssl_access.log combined
</VirtualHost>
```

PROYECTO 2º DAW - CURSO: 24 / 25

Una vez que apagamos y volvemos a encender el **servicio** de **Apache**, como se muestra en la captura, el **certificado SSL** se activa correctamente y pasamos a tener conexión **HTTPS** en **nuestra web**.



Configurar la base de datos (AURORA & RDS):

Lo primero que vamos a hacer es dirigirnos a esta sección y **crear la base de datos**.

The screenshot shows the AWS RDS Dashboard. On the left sidebar, under the 'Aurora and RDS' section, there are various options like 'Databases', 'Query editor', and 'Event subscriptions'. In the main content area, there's a blue banner at the top stating 'Introducing Aurora I/O-Optimized'. Below it, the 'Resources' section lists DB Instances (1/40), DB Clusters (0/40), and Snapshots (0). A large red circle highlights the 'Create a database' button, which is orange with white text. To the right of the resources, there are sections for 'Recommended services' (no recommendations yet) and 'Recommended for you' (links to AWS Backup, Cross-Region DR, and Aurora Global Database).

Una vez hemos hecho click ahí, **rellenaremos** los **datos** (poner el nombre de la base de datos, la contraseña, usuario)

Además deberemos de habilitar el acceso y poner el mismo en público al crearla.

Ya tan solo queda esperar a que se cree la base de datos, como vemos en la captura de pantalla que adjunto a continuación:

The screenshot shows the 'Databases' section of the AWS RDS dashboard. It displays a single database entry for 'musica-urbana', which is currently 'Backing...' and has an 'Instance' type of 'MySQL Co...'. The database is located in the 'us-east-1d' region and uses the 'db.t4g.micro' engine. The 'Size' is listed as 22.839 and the 'Current activity' is 0. At the top of the page, there's a note about 'Blue/Green deployment' and a 'Create database' button.

PROYECTO 2º DAW - CURSO: 24 / 25

IMPORTANTE: Una cosa **muy importante** que debemos de hacer es **habilitar** el **puerto** por defecto de **MYSQL** (el puerto por defecto es **3306**), para ello debemos agregar una regla y configurarla como se adjunta en la imagen a continuación:

sg-0b634c39026a2b1b3 - default

Details

Security group name: default
Owner: 56866394300

Security group ID: sg-0b634c39026a2b1b3
Inbound rules count: 2 Permission entries

Description: default VPC security group
VPC ID: vpc-0b2553cb9d78ad644

Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags

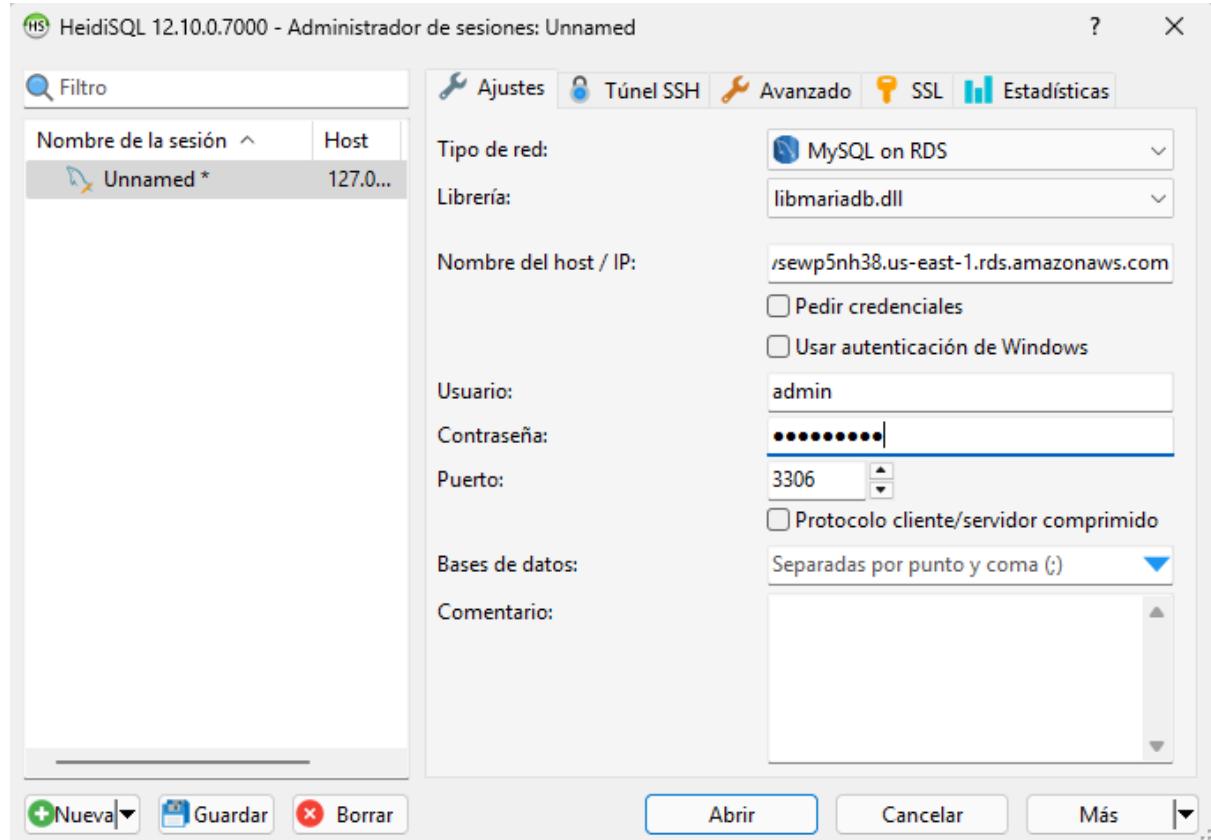
Inbound rules (2)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0ba892c0a873c643a	-	All traffic	All	All	sg-0b634c39026a2b1b...	-
-	sgr-0026a060d8c6fdfd6	IPv4	MySQL/Aurora	TCP	3306	0.0.0.0/0	-

Conexión a la base de datos (HEIDISQL):

Una vez tenemos **configurado** todo **como** en las **capturas anteriores**, debemos de **instalar** el siguiente [programa](#) para poder **gestionar** la **base de datos**.

Una vez instalado, tendremos una apariencia tal que así, la cuál deberemos de rellenar los parámetros que nos salen por pantalla, como se muestra en la imagen:



PROYECTO 2º DAW - CURSO: 24 / 25

Para poder saber la **IP** correspondiente de nuestro servidor, debemos de ir a nuestra **base de datos** en **AWS** alojada en **AURORA & RDS** y después hacer click en ella y se mostrarán los detalles correspondientes. En mi caso es así.

Connectivity & security

Endpoint & port

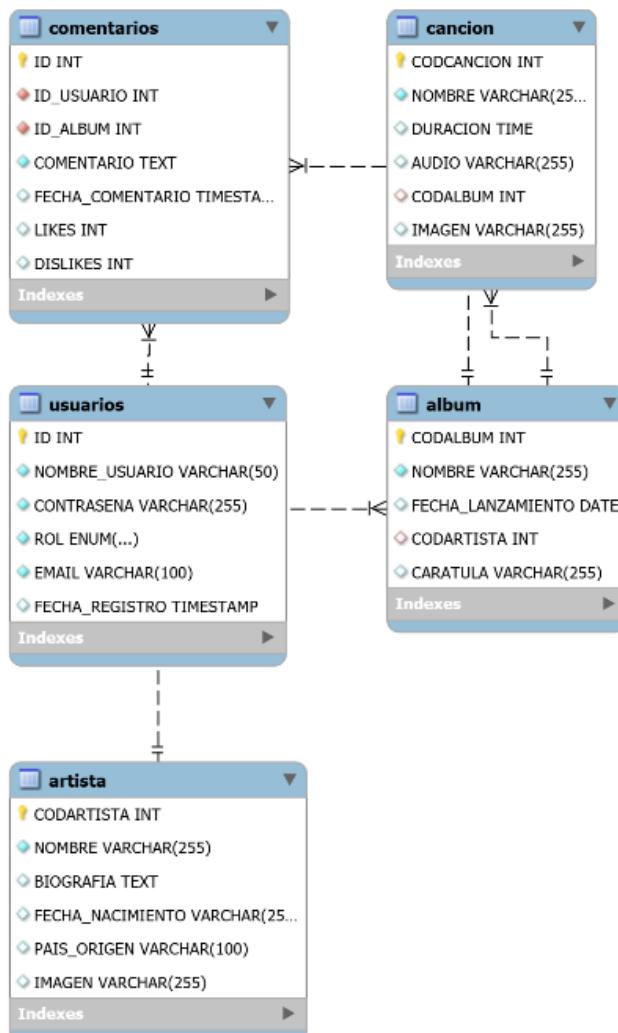
Endpoint

musica-urbana.cmwsewp5nh38.us-east-1.rds.amazonaws.com

Port

3306

Y ahora procederemos a crear la **base de datos**. En mi caso adjunte mi [**esquema de base de datos entidad relacional**](#).



Arquitectura del Sistema

En este apartado del documento proceré a explicar sobre el funcionamiento sobre una función seleccionada, en este caso veremos como ejemplo la función llamada **eliminarCancion**, explicaré el que hace exactamente, cómo se comunica, qué datos procesa, entre otros puntos que veremos. Bien, a continuación veremos el ejemplo:

¿Dónde está ubicada?

La función se encuentra en el archivo:

Ruta PHP: ./php/controladores/AdministrarControlador.php

```
function eliminarCancion($id): bool|string {
    global $db;
    try {
        // verificar que el id no este vacio
        if (empty($id)) {
            return respuesta(exito: false, mensaje: 'El ID de la canción es obligatorio.');
        }

        // obtener informacion de la cancion antes de eliminarla
        $stmt = $db->prepare(query: "SELECT NOMBRE, IMAGEN, AUDIO FROM CANCION WHERE CODCANCION = ?");
        $stmt->execute(params: [$id]);
        $cancion = $stmt->fetch(mode: PDO::FETCH_ASSOC);

        if ($cancion) {
            // crear rutas de archivos para eliminar
            $nombreArchivoAudio = '/assets/audio/' . $cancion['NOMBRE'] . '.mp3';
            $nombreArchivoImagen = '/assets/img/album/' . $cancion['NOMBRE'] . '.jpg';

            // eliminar archivo de audio si existe
            $rutaCompleta = $_SERVER['DOCUMENT_ROOT'] . $nombreArchivoAudio;
            if (file_exists(filename: $rutaCompleta)) {
                unlink(filename: $rutaCompleta);
            }

            // eliminar archivo de imagen si existe
            $rutaCompleta = $_SERVER['DOCUMENT_ROOT'] . $nombreArchivoImagen;
            if (file_exists(filename: $rutaCompleta)) {
                unlink(filename: $rutaCompleta);
            }

            // eliminar la cancion de la base de datos
            $stmt = $db->prepare(query: "DELETE FROM CANCION WHERE CODCANCION = ?");
            $stmt->execute(params: [$id]);

            return respuesta(exito: true, mensaje: 'Canción y archivos relacionados eliminados correctamente');
        } else {
            return respuesta(exito: false, mensaje: 'Canción no encontrada');
        }
    } catch (PDOException $e) {
        return respuesta(exito: false, mensaje: 'Error al eliminar la canción: ' . $e->getMessage());
    }
}
```

¿Cómo se llama desde el cliente?

La función **se llama desde el cliente mediante una petición fetch()** con **async/await** y cuando el usuario selecciona una canción cualquiera y después hace click en el botón para eliminarla.

Ruta JS: /assets/js/[ajax.js](#)

Nombre de función en JavaScript: eliminarCancion(event)

Método: POST con Content-Type: application/json

```
async function eliminarCancion(event) {
    event.preventDefault();

    const cancionId = document.getElementById('cancionEliminar').value;

    if (!cancionId) {
        mostrarToast('Por favor, selecciona una canción.', 'error');
        return;
    }

    try {
        const response = await fetch('./controladores/AdministrarControlador.php?accion=eliminar_cancion', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ id: cancionId })
        });

        const data = await response.json();

        if (data.exito) {
            mostrarToast('Canción eliminada correctamente', 'exito');
            llenarSelectCanciones();
        } else {
            mostrarToast('Error: ' + data.mensaje, 'error');
        }
    } catch (error) {
        console.error('Error en la solicitud:', error);
        mostrarToast('Error al eliminar la canción: ' + error.message, 'error');
    }
}
```

¿Qué hace exactamente?

La función lo que hace es básicamente, de la base de datos, coge y elimina el id de dicha canción y después elimina los archivos relacionados multimedia (el audio y la imagen).

```
if ($cancion) {
    // crear rutas de archivos para eliminar
    $nombreArchivoAudio = '/assets/audio/' . $cancion['NOMBRE'] . '.mp3';
    $nombreArchivoImagen = '/assets/img/album/' . $cancion['NOMBRE'] . '.jpg';

    // eliminar archivo de audio si existe
    $rutaCompleta = $_SERVER['DOCUMENT_ROOT'] . $nombreArchivoAudio;
    if (file_exists(filename: $rutaCompleta)) {
        unlink(filename: $rutaCompleta);
    }

    // eliminar archivo de imagen si existe
    $rutaCompleta = $_SERVER['DOCUMENT_ROOT'] . $nombreArchivoImagen;
    if (file_exists(filename: $rutaCompleta)) {
        unlink(filename: $rutaCompleta);
    }

    // eliminar la canción de la base de datos
    $stmt = $db->prepare(query: "DELETE FROM CANCION WHERE CODCANCION = ?");
    $stmt->execute(params: [$id]);
```

¿Qué archivos o partes del sistema involucra?

Frontend (HTML + JS):

- Select de canciones (<select> con ID cancionEliminar)
- Cuando el usuario le da click al botón, se activa la función eliminarCancion(event)

JavaScript:

- Archivo que ejecuta fetch() hacia el servidor y gestiona la respuesta (mostrar toasts, actualizar select).

PHP:

- Archivo AdministrarControlador.php contiene la lógica.

Base de datos:

- Tabla CANCION: se borra la canción con el id que se ha señalado a borrar.

Sistema de archivos del servidor:

- Se eliminan archivos multimedia (el audio y la imagen) relacionados a la canción.

JSON:

- Comunicación cliente-servidor se hace usando objetos JSON (tanto al enviar como al recibir).

¿Qué datos necesita y cómo se procesan?

La función necesita el **ID de la canción que se va a eliminar**, se **envían mediante fetch como JSON {id: cancion}** y la función de **PHP** se encarga de consultar los datos de la canción, se desvinculan los archivos del servidor usando **unlink()**, se borra el registro de la canción en la base de datos y después manda una respuesta el servidor al JSON, devolviendo un resultado u otro con un mensaje.

Fragmentos de Código:

Conexión a Base de Datos (PHP):

Este fragmento establece la conexión segura con la base de datos Aurora RDS en AWS. Utiliza PDO para mayor seguridad y manejo de errores.

```
<?php
require_once __DIR__ . '/config.php';

15 references
function obtenerConexion(): PDO
{
    try {
        $dsn = "mysql:host=" . DB_HOST . ";dbname=" . DB_NOMBRE . ";charset=" . CHARSET;
        $conexion = new PDO($dsn, username: DB_USUARIO, password: DB_CONTRASENA);
        $conexion->setAttribute(attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);
        return $conexion;
    } catch (PDOException $e) {
        throw new Exception(message: 'Error de conexión a la base de datos: ' . $e->getMessage());
    }
}
```

```
<?php
define(constant_name: 'DB_HOST', value: 'localhost');
define(constant_name: 'DB_NOMBRE', value: 'MUSICA_URBANA');
define(constant_name: 'DB_USUARIO', value: 'root');
define(constant_name: 'DB_CONTRASENA', value: 'root');
define(constant_name: 'CHARSET', value: 'utf8mb4');
```

Control de Roles y Sesiones (PHP):

Controla el acceso según el rol del usuario (admin/usuario normal). Los administradores pueden gestionar contenido mientras que los usuarios regulares solo pueden reproducir música y navegar.

```
session_start();
$_SESSION['id_usuario'] = $usuario['ID'];
$_SESSION['nombre_usuario'] = $usuario['NOMBRE_USUARIO'];
$_SESSION['rol'] = $usuario['ROL'];

setcookie(name: 'sesionActiva', value: 'true', expires_or_options: time() + (7 * 24 * 60 * 60), path: "/");
setcookie(name: 'usuarioNombre', value: $usuario['NOMBRE_USUARIO'], expires_or_options: time() + (7 * 24 * 60 * 60), path: "/");
```

Inicio de sesión (Fetch API):

Sistema de autenticación asíncrono que valida credenciales, maneja cookies para persistencia de sesión y actualiza la interfaz dinámicamente según el rol del usuario (admin/normal). Utiliza async/await para mejor manejo de promesas y control de errores.

```
try {
    // crear un objeto formData para enviar los datos al servidor
    var formData = new FormData();
    formData.append('nombreUsuario', nombreUsuario);
    formData.append('contrasena', contrasena);

    var respuesta = await fetch('./controladores/LoginControlador.php', {
        method: 'POST',
        body: formData
    });

    var datos = await respuesta.json();

    if (datos.exito) {
        mostrarToast("Inicio de sesión exitoso", 'exito');

        // guardar información en cookies para recordar la sesión del usuario
        crearCookie('sesionActiva', 'true', 7);
        crearCookie('usuarioNombre', nombreUsuario, 7);

        // verificar si el usuario es administrador y guardar esta información en una cookie
        if (datos.esAdmin) {
            crearCookie('esAdmin', 'true', 7);
        } else {
            crearCookie('esAdmin', 'false', 7);
        }
    }
}
```

Resolución de problemas:

Problema 1:

Me encontré un problema al configurar el enviar correos electrónicos mediante la **librería de PHPMailer**, me encontraba el mismo error: SMTP Error: Could not authenticate.

PHPMailer/PHPMailer

**#1025 SMTP Error:
Could not authenticate**

Solución aplicada 1:

Para solucionar este problema, lo primero que hice fue investigar por internet, sobretodo a través de [stackoverflow](#), y observé que era un problema bastante común y para solucionarlo lo primero que había que hacer era dirigirnos a Google, a nuestra cuenta que vayamos a usar para esta librería de PHPMailer.

Debemos de ir a la sección de contraseñas de aplicaciones y crear una nueva, y con esa nueva contraseña de aplicación ya sí podríamos utilizarla en nuestro aplicativo.

[Tus contraseñas de aplicación](#)

PHPMAILER-PROYECTO-FINAL

Creada: 8 feb; utilizada por última vez:
0:37



Problema 2:

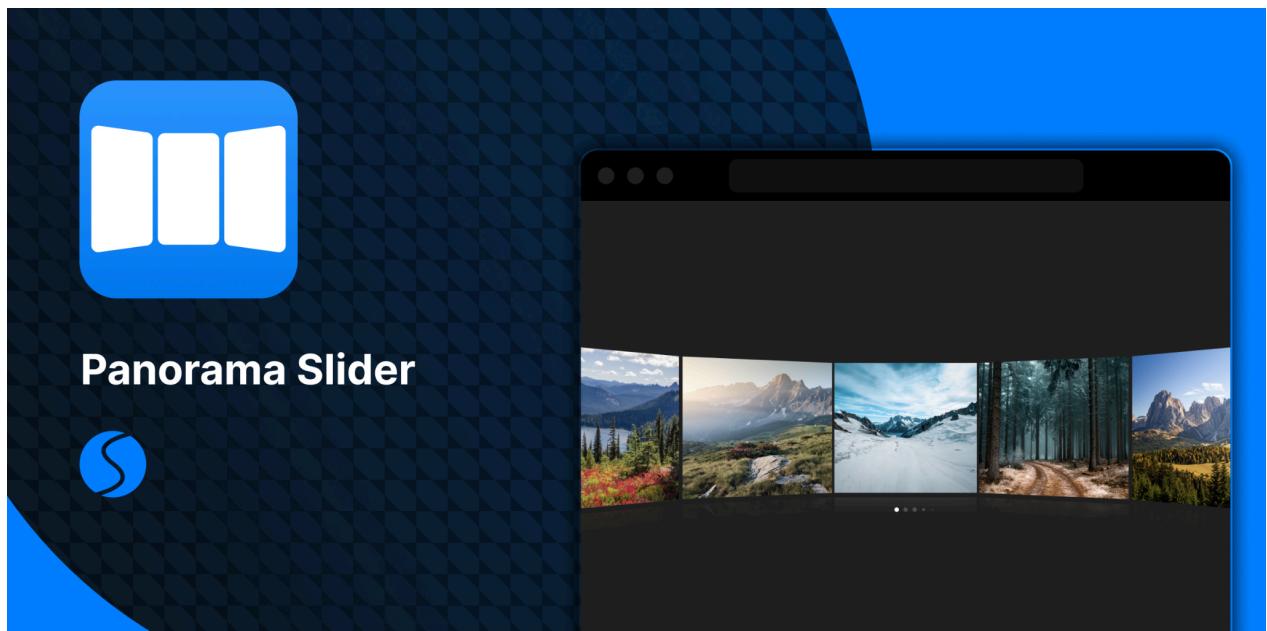
Los álbumes y artistas de la página.

Al principio cuando empecé a realizar el diseño de la página web, no tuve en cuenta en primero crear un carrusel que sea responsive y se pudiera seleccionar el álbum que sea, por lo que pensé en alternativas para poder realizar un carrusel de manera óptima para mi página sin consumir muchos recursos, por lo que opté por utilizar [Swiper.js](#).

Solución aplicada 2:

Investigación por internet de posibles candidatos que ofrezcan un rendimiento para mi página web, además de que sea responsive tanto para móviles, tablets y ordenadores.

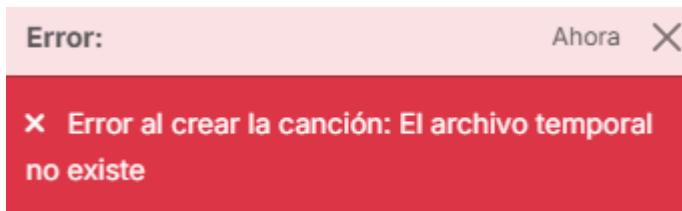
Tenía varios candidatos, entre ellos estaba **Slick Slider** y **Owl Carousel**, pero estos, ambos **tenían un problema** y es que eran **muy pesados, muy poco optimizado** para los dispositivos móviles, **menos personalizables**, etc ...



Problema 3:

Cuando intenté añadir una nueva música a mi página web desde el apartado de gestionar la página web siendo administrador, me encontré un problema.

El problema era que no estaba asignada la carpeta temporal en php.



Solución aplicada 3:

Investigando por internet, sobre todo en varios foros, específicamente busqué y encontré en **stackoverflow**, a través de este [enlace](#), de que cuando **no hay un directorio temporal definido o si el servidor no tiene permisos para usarlo**, PHP **no puede guardar el archivo durante la subida** y por eso falla el proceso y el sistema necesita un espacio temporal en el servidor donde guardar ese archivo mientras se procesa la subida.

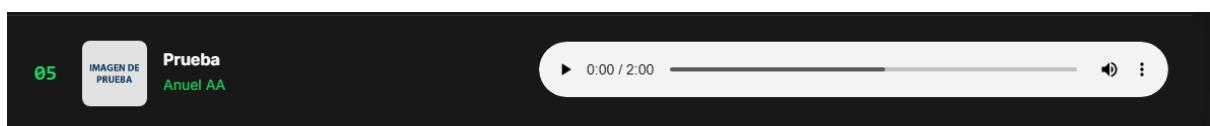
Este **problema** se debe a que el **archivo php.ini no tiene configurado los archivos temporales ni los permisos suficientes**.

```
[ec2-user@ip-172-31-28-101 ~]$ php -i | grep upload_tmp_dir  
upload_tmp_dir => no value => no value  
[ec2-user@ip-172-31-28-101 ~]$ █
```

Nos dirigimos a **php.ini** y **modificamos la siguiente línea para poder tener los archivos temporales**, además **le damos permisos a la carpeta y listo**, ya tendríamos funcionando el apartado de gestionar las canciones correctamente.

```
; Temporary directory for HTTP up  
; specified).  
; https://php.net/upload-tmp-dir  
upload_tmp_dir = /tmp█
```

Ejemplo de prueba de que funcionó:



Pliego de Condiciones:

Requisitos funcionales:

- El usuario puede registrarse y también puede iniciar sesión.
- El usuario puede ver los álbumes y los artistas, también en cada álbum están las canciones.
- El usuario que es admin puede subir, modificar y eliminar canciones, álbumes y artistas.
- El usuario tiene una barra de búsqueda, dónde puede buscar canciones, álbumes y artistas.
- El usuario puede ver su perfil, su nombre ,su rol, email y cuando se registró en la página.

Requisitos técnicos:

- El sistema está desplegado en una [instancia EC2 de AWS](#).
- La base de datos está en [Aurora & RDS AWS](#).
- El sitio usa HTTPS con un [certificado SSL instalado con Certbot](#) para el [dominio](#) y para la [IP elástica](#) con [Apache](#).
- Toda la web es responsive gracias a CSS y Bootstrap.
- Se usan tecnologías como Swiper.js y JavaScript ES6.
- El repositorio del proyecto está en [GitHub](#) y contiene toda la documentación y el código fuente.

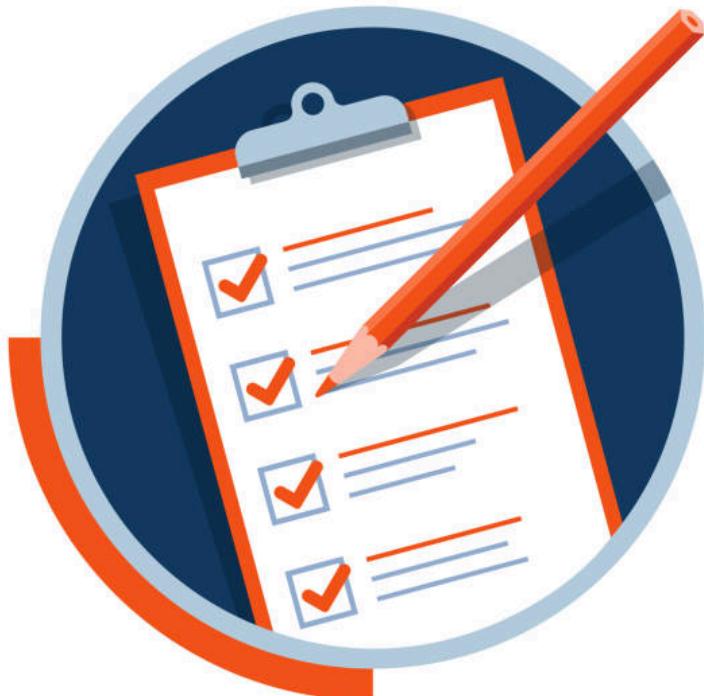
URL Aplicación desplegada:

La **aplicación**, podremos **encontrarla** de dos maneras:

- A través del [dominio gratuito que usé](#), **accesible** a través de este [enlace](#)
- A través de la **dirección IP elástica accesible a través del siguiente enlace**.

Bitácora de tareas realizadas:

Se ha realizado una **bitácora de tareas** en **hojas de cálculo de google**, para poder acceder a ella, tan solo basta con acceder a través de este [enlace](#).



Participación de usuarios:

He recopilado una serie de **pruebas que les hice a mis familiares y amigos sobre el funcionamiento de mi página web**, para que ellos me pudiesen **aportar posibles mejoras, errores que se han encontrado**, además me han aportado feedback positivo para poder mejorar la aplicación en un futuro.

Aportaciones positivas:

- El diseño era moderno, bonito, llamativo, bastante colorido, etc.
- Funcionamiento correcto de los álbumes, se puede reproducir música, se ven los artistas.
- Se pueden realizar búsquedas y dar los mismos resultados en la barra de búsqueda.
- Aspecto responsive en teléfonos móviles, tablets, ordenadores y portátiles.
- Estética, transiciones, usabilidad fácil.

Aportaciones negativas:

- Sugerencia de que el menú desplegable fuera más pequeño.
- No tiene un aspecto tan único, ya que me basé en el de Spotify.
- No contiene mucho contenido textual, en la página de inicio.

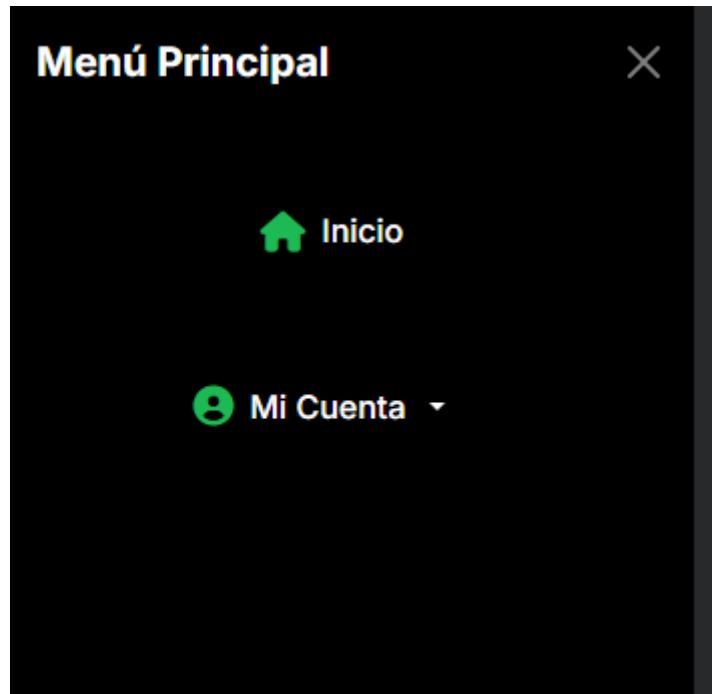


Conclusiones y planes a futuro:

En este apartado, considero varios **cambios** a futuro sobre **mi aplicación web**, a continuación muestro algunos cambios sugerentes:

Mejorar diseño y adaptación a móviles y tablets:

Me gustaría en un futuro **mejorar** este apartado en mi página web, ya que en **diseño** actual, funciona pero tiene algunos aspectos que mejorar, como por ejemplo, **el menú desplegable** (el menú hamburguesa), al seleccionar en móviles te ocupa toda la pantalla y para moverte en mi página web debes de seleccionar la página y hacer click en el aspa (la 'X') para cerrar el menú para poder ver el contenido de la página.



Añadir más contenido para mejorar mi web:

Otro planes a futuro que me gustaría es incorporar más contenido, tales como textos, párrafos, títulos y otros componentes, porque es importante para optimizar el [SEO](#) de mi sitio web. Esto facilita que los **motores de búsqueda comprendan mejor el contenido de la página y la presenten a un mayor número de individuos interesados.** Además, un contenido más completo y variado hace que los visitantes pasen más tiempo navegando, lo que mejora la experiencia del usuario en el sitio.



Ofrecer opciones de personalización para los usuarios:

Una idea importante para el futuro de mi web es darle a los usuarios la opción de **personalizarla** a su gusto. Eso quiere decir que cada usuario pueda **cambiar** cosas como los colores, el tamaño de las letras, o elegir qué contenido quiere ver primero.

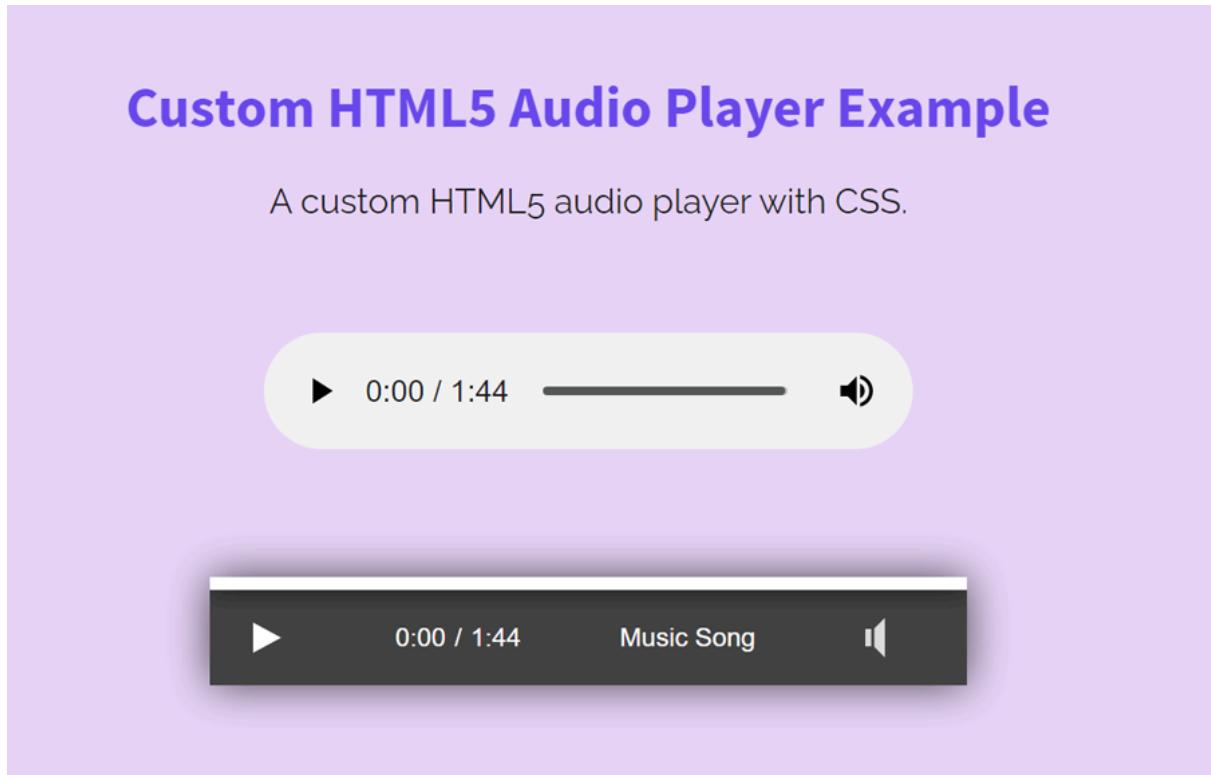
Eso hace que quieran volver más seguido. Ofrecer estas opciones, también ayudaría a que mi web destaque entre otras, y se vea más profesional, pero a la vez cercana.



Mejorar la interfaz de reproducir música:

Otra idea que tengo de **planes a futuro** para **mejorar mi página web**, es el **diseño de los multimedia**, concretamente, los audios, ya que estos, tienen la apariencia por defecto y me gustaría aplicarle un **diseño que sea más moderno, minimalista y sencillo para el usuario.**

Me gustaría poder aplicar un diseño parecido al adjuntado en la imagen a continuación.



Video de demostración:

A continuación, **adjunto video**, en el cuál se muestra el funcionamiento de la página web, demostrando su funcionalidad, su interfaz, el cómo se adapta automáticamente según el dispositivo (**responsive**), punto de vista de los roles (**administrador** y **usuario**), **gestionar la página web** (funcionamiento mostrado uno por uno), etc.



Bibliografía utilizada:

HTML: [Enlace 1](#), [Enlace 2](#), [Enlace 3](#), [Enlace 4](#).

CSS: [Enlace 1](#), [Enlace 2](#), [Enlace 3](#).

Bootstrap: [Enlace 1](#).

Javascript: [Enlace 1](#), [Enlace 2](#), [Enlace 3](#), [Enlace 4](#), [Enlace 5](#), [Enlace 6](#), [Enlace 7](#).

FontAwesome: [Enlace 1](#), [Enlace 2](#).

Swiper.js: [Enlace 1](#), [Enlace 2](#).

PHP: [Enlace 1](#), [Enlace 2](#).

Apache: [Enlace 1](#).

MYSQL: [Enlace 1](#).

AWS: [Enlace 1](#), [Enlace 2](#), [Enlace 3](#), [Enlace 4](#), [Enlace 5](#).

