

Final Aplicaciones para Dispositivos Móviles

Alumno: Mateo D. Neglia

Profesora: Mabel García



Para esta aplicación se creó un concurso ficticio de una radio en el que las personas suben una reseña de entre 10 álbumes destacados semanales y cada lunes se elige una reseña ganadora que se lleva un premio semanal de boucher de comida o algún descuento en alguna compra seleccionada y tanto el premio como los 10 discos cambian a cada semana.

El concurso se llama “Listener’s Choice” (Elección del oyente), la página fue realizada enteramente en Vue 2.6 (como fue pautado en la consigna), bootstrap, y se usaron elementos de animación nativos de vue además de la librería de Vuetify para varios componentes del sitio. También este sitio emplea una base de datos creada para él mismo así también como la concedida por la API de Spotify.

App.vue y Navbar:

Se está usando b-navbar de bootstrap para Vue en combinación con router-view para el enrutado de la página. Para cambiar las vistas también se está usando una transición de Vue de fade-in left. La barra de navegación tiene dos ítems que escuchan a una variable “estaLogIn” con un v-if en cada uno para mostrar la página del inicio de sesión si el usuario no está logueado y la página de detalles de usuario si este sí está logueado. Esta variable junto a otra (usuarioLogeado) dependen ambas del estado actual del sessionStorage, cambiando de esta forma parte de lo que ve el usuario en la barra de navegación.

LandingComponent(Home):

Este es el componente que funciona como la “home” del sitio. Aquí hay un baner que cambia sus dimensiones y diseño dependiendo de la vista del usuario (mobile, tablet, desktop). Debajo, hay una serie de cards con imágenes diseñadas especialmente para la página. Se usó para esta estructura un modelado de v-container de Vuetify con varias v-card, v-img, v-actions, y v-btn. Para la sección de debajo de todo se armó un array de objetos con los links de las imágenes customizadas para el sitio y varios ítems de cada tarjeta. Para la distribución de estos datos se usó un v-for y cada una de las tarjetas tiene un botón que apunta a un sector específico del sitio para informar a nuevos usuarios que lo estén navegando.



AboutUs(Sobre Nosotros):

Este componente es el sobre nosotros de la página, es bastante simple comparado con los otros ya que por su función en sí no requiere demasiada interacción con el usuario viendo que su propósito es puramente informativo. Aquí se resaltan con strong segmentos de información relevantes al uso de la página y al final hay un v-btn de Vuetify para redirigir al Usuario al formulario de reseña del sitio.



MusicForm(Escribe tu reseña):

Este es el componente principal del sitio. Hay varios comportamientos que están siendo encadenados ni bien la aplicación es dirigida a este. A nivel visual, primero el usuario ve el v-skeleton-loader de Vuetify antes de que la información que usa este sea cargada en el formulario. Cuando el componente es montado se llama a dos funciones, una que es `fetchAlbumData` y otra que es `getSessionData`. La función `fetchAlbumData` nos trae información del servicio de spotify el cual lo estamos llamando desde un archivo javascript en un directorio que dedicamos a los servicios de la aplicación.

Sobre el servicio: Aquí se hacen dos llamadas a la api de spotify, una para conseguir un token con el cual se hacen las comunicaciones con spotify desde el sitio y otra para traer una lista de álbumes con sus datos. Para traer la lista de álbumes también se llama a un servicio custom hecho para este sitio que devuelve una lista de strings con ids de los álbumes que requiere el servicio que trae los datos de los álbumes. Entonces `"getAccessToken"` nos trae el token para comunicarnos con la API de spotify y `"getAlbumData"` llama primero a `"getAlbumIds"` para traer los ID para la llamada a spotify y luego realiza la comunicación con los resultados tanto del token como del string de IDs separados por comas como está estipulado en la documentación de la API.


Volviendo al componente, cuando es montado llama también a una función llamada `"getSessionData"` cuyo propósito es informar al componente si tiene que cargar algunos ítems del formulario o no dependiendo si hay un usuario actualmente logueado o no. Si hay un usuario logueado, los componentes para el nombre o email no son cargados y si no hay un usuario logueado carga los input de email y nombre.

Una vez resuelta esta vista, se carga el formulario compuesto por el nombre del reseñista, su email, un dropdown con los discos disponibles para reseñar. Al seleccionar un disco, abajo se carga una imagen de este la cual se adapta dependiendo de la vista(desktop, tablet, mobile). Debajo hay una lista de "rangos" con los cuales se puede elegir el disco que van desde cobre, bronce, plata, oro, diamante. Y debajo de esto está el text-area para escribir la reseña. Luego hay un par de preguntas seleccionables por "sí" o "no" y una última para seleccionar si el usuario quiere recibir mails con recomendaciones musicales.

Cada uno de los datos de este formulario están asignados a una variable mediante el uso de v-model para enviar dinámicamente los datos de este cuando el usuario lo complete.

De igual manera, también se armó un sistema de validaciones para los elementos del formulario que son requerimientos de compleción que muestran un mensaje de error al usuario si hay algún error de validación que impida el envío de los datos.

Si los datos pasan la validación, estos son agregados a un array el cual luego es guardado en localStorage. Cuando el usuario completa los campos, estos se limpian y el usuario recibe una notificación de que su formulario fue enviado y debajo del formulario aparecen las últimas reseñas del usuario si este tiene alguna. Todos los inputs de este formulario tienen una lógica ternaria que cambia el estilo del mismo si el input o text-area tiene contenido o no.




[HOME](#)
[SOBRE NOSOTROS](#)
[ESCRIBE TU RESEÑA](#)
[RESEÑAS](#)
[ÁLBUMES DE LA SEMANA](#)
[ADMIN](#)


Envíanos la reseña de tu álbum


Selecciona un álbum y envía tu reseña!


Selecciona tu Disco


Elige el rango del rango del Álbum:

 Cobre

 Bronce


 Plata


 Oro

 Diamante


Escribe aquí tu reseña


¿Comprarias digitalmente este álbum?

 Si

 No

¿Escucharías más álbums de este género?

 Si

 No

Marca este campo si quieres recibir mails con recomendaciones musicales: ☐

Enviar Reseña

Aún no tenemos datos. Completa tu reseña y envíala!!

UserRankings(Reseñas):


Este componente está dividido en dos secciones, una para las reseñas del usuario navegando y otra para reseñas cargadas de otros usuarios. Las reseñas del usuario que está navegando corresponden a las guardadas en localStorage y las de otros usuarios están siendo traídas por una api custom con reseñas cargadas en la base de datos, de esto se ocupa el servicio “review-service” que hace la consulta por fetch. Al traer las reseñas de la DB se las asigna a un array y este array reparte los datos en v-cards de Vuetify mediante un v-for.

Similarmente se carga un segundo array pero con los datos de localStorage en la sección de reseñas del usuario, hay dos secciones dentro de esta separadas por un condicional v-if que depende si el array de reseñas del usuario está con elementos o no, si los tiene los carga y si no los tiene muestra un mensaje con un botón para que el usuario vaya a cargar una reseña (usando el componente de Vuetify v-btn).

La diferencia crucial de las tarjetas del usuario(en localStorage) a las de otros usuarios(en DB) además de dónde se nutren para mostrar sus datos es que las tarjetas del usuario de localStorage tienen dos botones que representan la sección B(borrar) y M(modificar) del ABM principal del sitio(siendolo el formulario de crear una reseña principal el A(Alta)).

Apretando editar, somos llevados al formulario de edición de la reseña y apretando borrar borramos la reseña sobre la cual clickeamos. Además de estos botones, hay un borrado masivo en un botón de abajo de todo de esta sección que nos permite eliminar todas nuestras reseñas.

Al ir a editar la reseña, pasamos el ID del disco reseñado que nos va a servir para hacer las modificaciones pertinentes en la pantalla de edición de la reseña.

[HOME](#) [SOBRE NOSOTROS](#) [ESCRIBE TU RESEÑA](#) [RESEÑAS](#) [ÁLBUMES DE LA SEMANA](#) [ADMIN](#)

Estas son tus reseñas:

Matt

Álbum Seleccionado: **Thick as a Brick**

Álbum rankeado como: Diamond

Y la reseña de Matt fue: Thick as a brick es excelente

[EDITAR](#) [BORRAR](#)

[BORRAR TODAS LAS RESEÑAS](#)

Estas son las actuales reseñas del sitio:

David

Álbum Seleccionado: **Thick as a Brick**

Álbum rankeado como: Gold

Y la reseña de David fue: ¿Qué más puedo decir sobre esta obra maestra? Este disco me introdujo a la flauta traversa en el rock cuando era chico e Ian Anderson es uno de los pilares del Rock Progresivo, sinceramente una calidad única y una joya que merece más reconocimiento.

Matt

Álbum Seleccionado: **Thriller**

Álbum rankeado como: Diamond

Y la reseña de Matt fue: Este fue el álbum que me hizo amar el pop, ¿ya no lo hacen como antes no? Si bien la figura pública de MJ fue controversial a lo largo de los años, veo a esta creación como un antes y un después en la música moderna.

Juan

Álbum Seleccionado: **Metallica(Remastered 2021)**

Álbum rankeado como: Silver

Y la reseña de Juan fue: Creo que este fue el disco que convirtió a todo el desconfiado del género del metal a un fiel seguidor en su momento, esta es una puerta de entrada a muchas otras grandes bandas.

MusicFormEdition.vue(Edición de Reseña):

Aquí el comportamiento es parecido al del formulario con algunas diferencias claves sobre lo que el usuario puede editar. Primero, usamos el id que pasamos por params para hacer una llamada al servicio de spotify pero esta vez sin tener el servicio intermediario de la lista de ids (le estamos pasando un id específico, no necesitamos más data que esa). Usamos la función “fetchSpotifyAlbum”, la cual es una función que está siendo reutilizada dentro del mismo javascript que maneja el servicio para evitar la duplicación de código.

Al llegar los datos, se carga el formulario que nos muestra el nombre del álbum de la reseña y nos muestra una portada, además, vemos el resto de los inputs, radio, y text-areas sin incluir edición para el nombre de usuario, email y selección de álbum.

Luego, la lógica es similar a la del formulario de alta para la reseña con la diferencia que luego de editar la reseña somos redirigidos a la página de las reseñas donde podemos ver los cambios de forma resumida.

Aquí puedes editar la reseña de Thick as a Brick



Elige el rango del rango del Álbum:

- ☐ Cobre
☐ Bronce
☐ Plata
☐ Oro
☐ Diamante

Thick as a brick es excelente

¿Comprarias digitalmente este álbum?

- ☒ Si ☐ No

¿Escucharías más álbums de este género?

- ☒ Si ☐ No

Marca este campo si quieres recibir mails con recomendaciones musicales: ☒

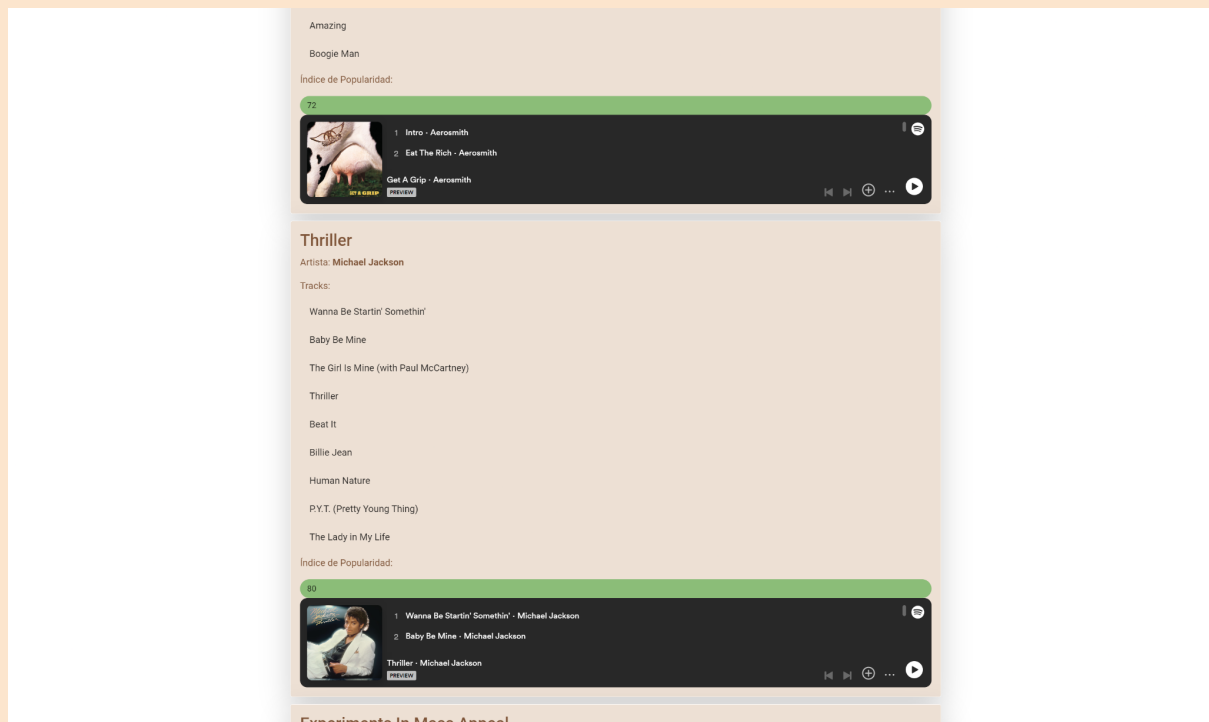
[Editar Reseña](#)

AlbumComponent(Álbumes de la Semana):

Esta es una sección de la página donde el usuario puede navegar por los Álbumes que hay disponibles en la semana. Cuando el componente es montado, busca la lista de álbumes de la misma forma que lo hace el formulario principal de la página pero utiliza los datos de los álbumes de otra forma.

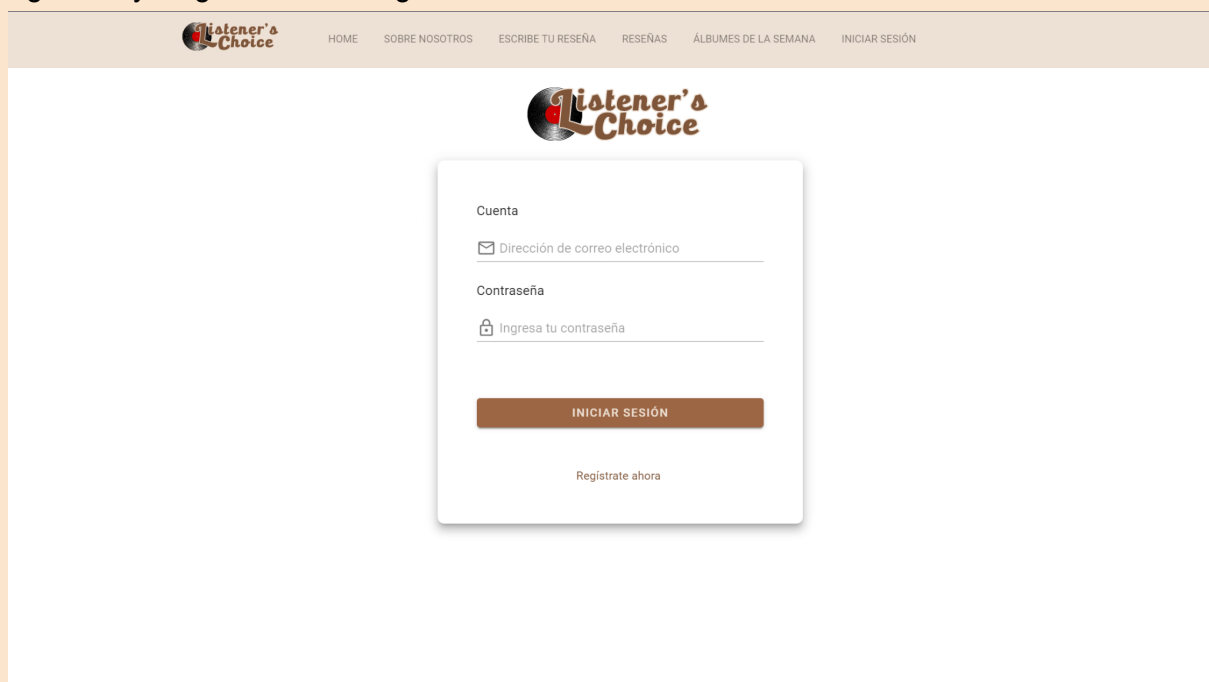
Se hace un v-for con estos datos donde se crean varias v-cards de Vuetify donde se muestran los datos de: Nombre del artista, Nombre del disco, Lista de canciones, índice de popularidad y se carga un Iframe con todos los temas del disco donde se puede escuchar desde la página partes de los temas antes de escribir una reseña y dar un veredicto.

Hay dos comportamientos desarrollados para estas tarjetas en particular. Uno de ellos es el usar el componente v-chip (Vuetify) dentro de las v-list (Vuetify). Los v-chip en este caso muestran el valor “Explícito” si el tema tiene letras o contenido no apto para todo público. Además se armó una función que recibe un valor numérico del índice de popularidad de cada disco y muestra este índice en un chip que varía su color dependiendo de qué tan popular es el disco.



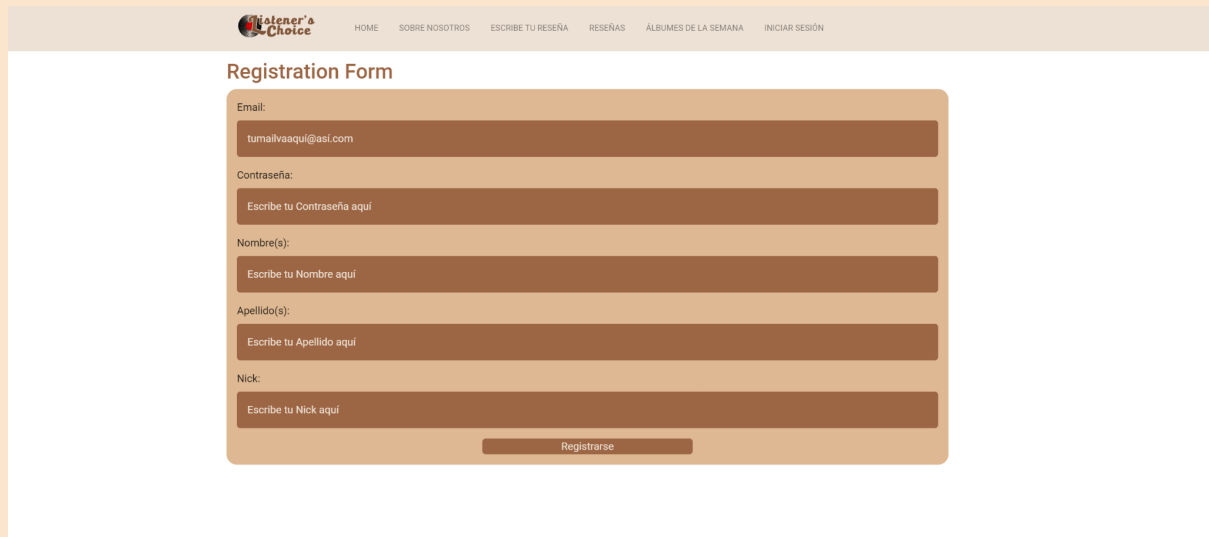
LogIn(Inicio de sesión):

Aquí uso un componente compuesto de Vuetify el cual adapté para los estilos de la página para hacer un log in y redirección a registro de usuario si este no tiene aún una cuenta. Para iniciar la sesión se usa una llamada por fetch a la base de datos con los usuarios mandándole por post un email y la contraseña, si estos matchean con los datos manejados por la lógica desde php, recibimos los datos del usuario los cuales son asignados a un sessionStorage junto con una variable que le indica a la aplicación que hay un usuario logueado y luego somos redirigidos a la home.



Registro

El registro hace una llamada a la lógica de php donde le mandamos por post los datos ingresados por el usuario y hace un insert en la tabla con los datos además de hashear algunos elementos sensibles como la password.



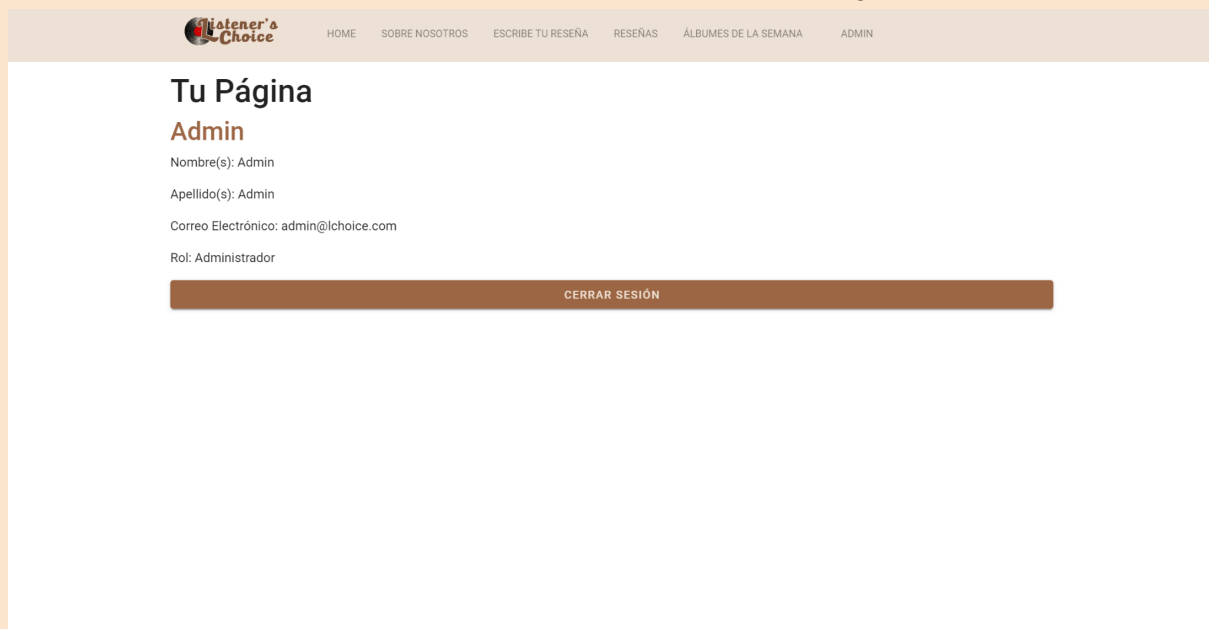
The screenshot shows the 'Registration Form' on the website. The form is titled 'Registration Form' and contains several input fields for user registration. The fields are labeled as follows:

- Email:
- Contraseña:
- Nombre(s):
- Apellido(s):
- Nick:

At the bottom of the form is a button labeled 'Registrarse'.

Perfil de Usuario:

El perfil de usuario trae los datos no sensibles del sessionstorage y se los muestra al usuario conectado: Nombre, Apellido, Nickname, Email y Rol. Como el rol viene como un número, usamos una función para convertir el número en un string con el rol.



The screenshot shows the 'Tu Página' (Your Page) for an Admin user. The page displays the following information:

- Tu Página**
- Admin**
- Nombre(s): Admin
- Apellido(s): Admin
- Correo Electrónico: admin@lchoice.com
- Rol: Administrador

At the bottom of the page is a button labeled 'CERRAR SESIÓN'.