Escuela de Ingenierías
Industrial, Informática y Aeroespacial

universidad de león

Luis Flórez Juárez

Unileón

13/01/2021

Índice

1.Desarrollo del problema

Actualmente la música electrónica es un género de música bastante conocido mundialmente, pero la mayoría de personas, salvo aquellas personas que les gusta este género de música, no conocen los diferentes subgéneros que este tipo de música tiene.

Estos distintos subgéneros pueden hacer que las canciones sean muy diferentes unas de otras y también cambiar la forma en la que vivimos esa música.

Para solucionar este problema me decidí a desarrollar ElectroHits que es una página de reomendacion de música electrónica. En ella el usuario puede obtener una recomendación aleatoria entre todas las canciones o una recomendación basada en el subgénero que el usuario haya escogido previamente.

2.Herramientas utilizadas

Mi aplicación está desarrollada con Vue.js que está basado en JavaScript y conectada a una base de datos local con node4j la cual utiliza grafos. La página web está alojada como una aplicación de escritorio utilizando electron.js.

Neo4j

Usado para crear la base de datos.

Para crear esas bases de datos utiliza grafos.

Las bases de datos están diseñadas para tratar las relaciones entre los datos con la misma importancia en la que se tratan los datos en sí. Está destinado a contener datos sin restringirlos a un modelo predefinido.[1]

Estas bases de datos están compuestas por dos elementos:

- Los nodos (vértices): representan las entidades del grafo. Pueden contener cualquier número de atributos (también llamados propiedades). Los nodos se pueden etiquetar con labels que representan sus diferentes roles en su dominio.
- Las relaciones (aristas): proporcionan conexiones dirigidas y nombradas semánticamente relevantes entre dos nodos. Una relación siempre tiene una dirección, un tipo, un nodo inicial y un nodo final. Como los nodos, las relaciones también pueden tener propiedades.

Para realizar las consultas a la base de datos se utiliza un lenguaje llamado Chypher. Este lenguaje se utiliza para crear la base de datos.

Vue.js

Es un framework progresivo para construir interfaces de usuario. Está diseñado desde cero para ser utilizado incrementalmente. Esta librería está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes. [2]

JavaScript

Es un lenguaje de programación ligero, interpretado o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Es más conocido como un lenguaje de scripting para páginas web y es usado en muchos entornos fuera del navegador como Node.js. [3]

Es el lenguaje que he utilizado para programar la página web.

Node.js

Node.js es un entorno de código abierto pensado para hacer posible el desarrollo de aplicaciones JavaScript del lado del servidor. Lo que permite que una vez desarrolladas puedan ejecutarse dentro de diferentes sistemas como Microsoft Windows o Linux.

Este entrono también proporciona una nutrida biblioteca con módulos de JavaScript que simplifica el desarrollo de aplicaciones web usando este framework en gran medida. [4]

Vuetify

Vuetify es un framework que combina la potencia de Vue.js con la estética de Material Desing. Permite acelerar el desarrollo de aplicaciones web complejas, incorporando una gran cantidad de componentes “listos para usar”. [6]

Lo he utilizado para crear la interfaz de la aplicación.

3.Desarrollo de la aplicación

Base de datos:

Es una base de datos de grafos que contiene 155 nodos y 154 relaciones. La base de datos esta creada de la siguiente forma:

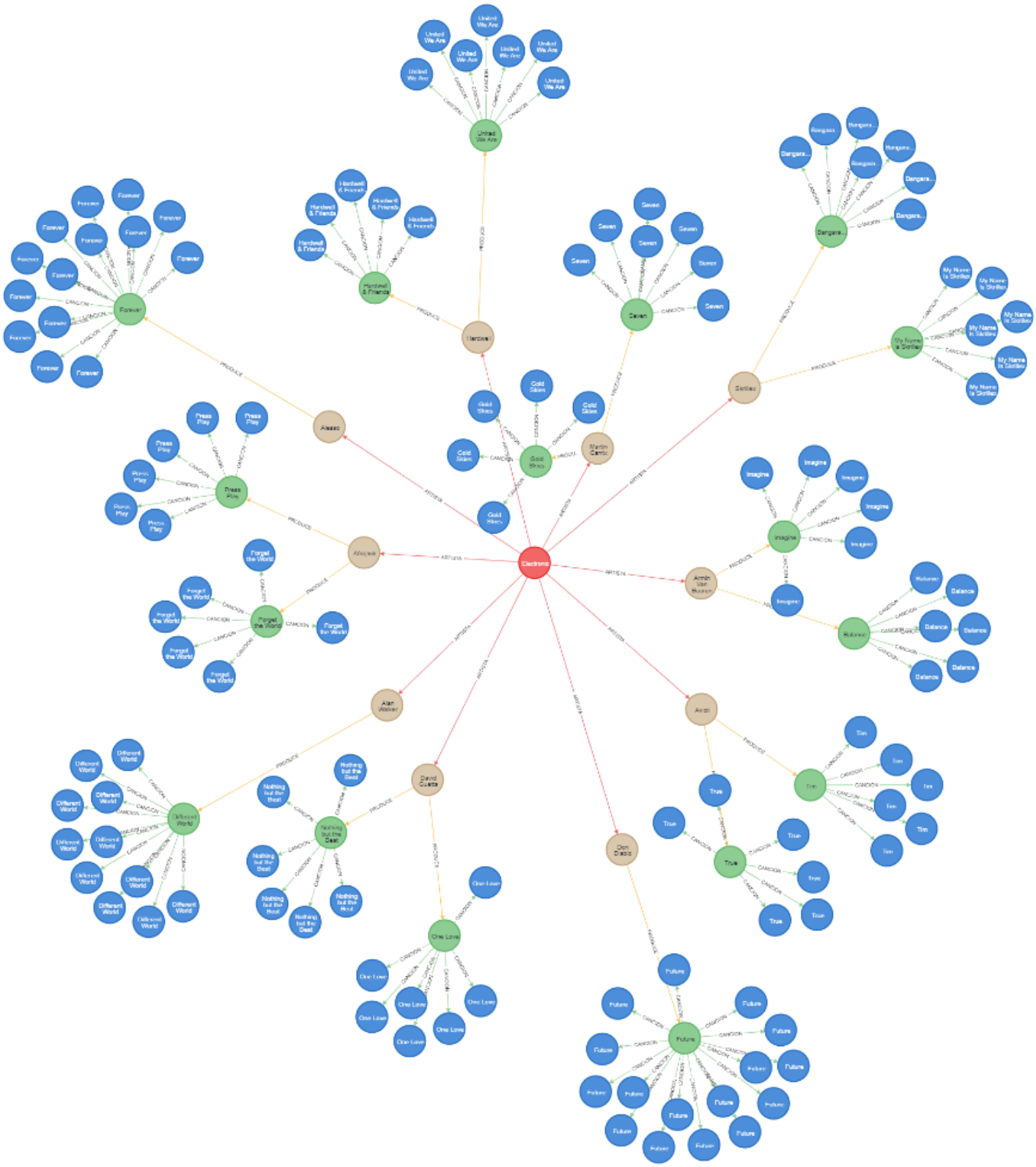
Genero ⟶ Artistas ⟶ Álbumes ⟶ Canciones

Cada uno de estos nodos tiene varias propiedades:

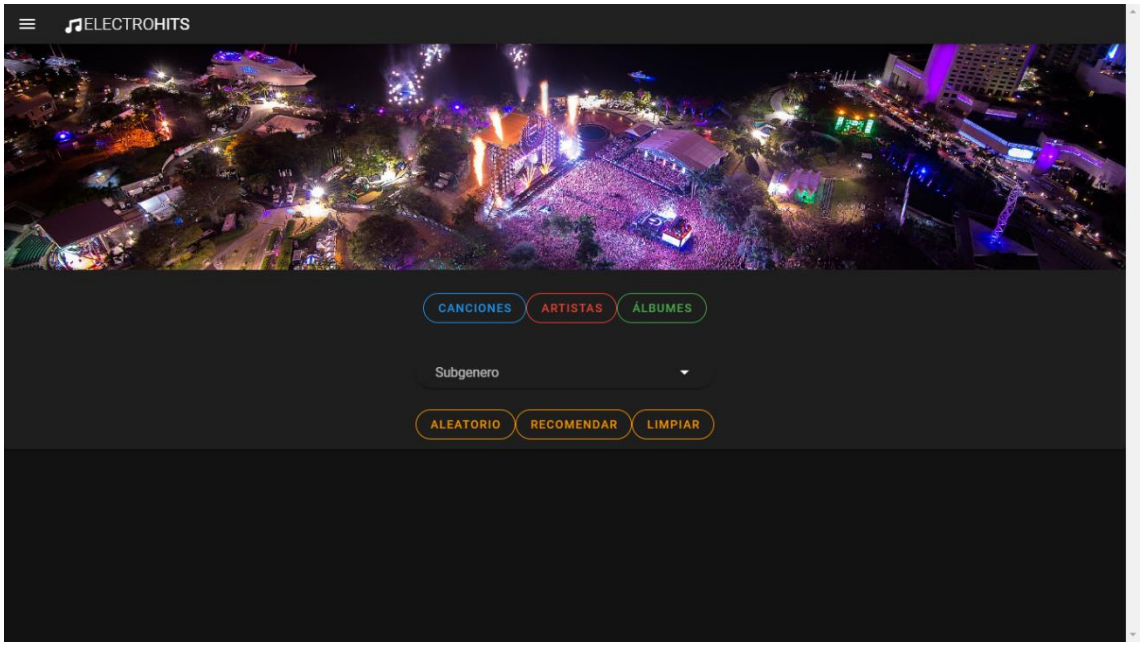
- Género: name
- Artistas: name, from, born
- Álbumes: name, released, group
- Canciones: name, released, duration, album, trackNumber, group, subgender

Las relaciones de la base de datos también tienen propiedades:

Género – [ARTISTA] -> artista – [PRODUCE] -> álbum – [CANCIÓN] -> canciones


Electronic
ARTISTA
PRODUCE
CANCION
United We Are
Hardwell
Hardwell & Friends
Forever
Seven
Dangara...
Skrillex
My Name is Skrillex
Martin Garrix
Gold Skies
Avicii
Imagine
Balance
Armin Van Buuren
Alesso
Fresa Play
Afrojack
Forget the World
Alan Walker
Different World
David Guetta
Nothing but the Beat
One Love
Don Diablo
Future
True
Tim

Interfaz de la aplicación



La interfaz básica está compuesta por una imagen, 6 botones y un combobox para seleccionar el subgénero de música electrónica.

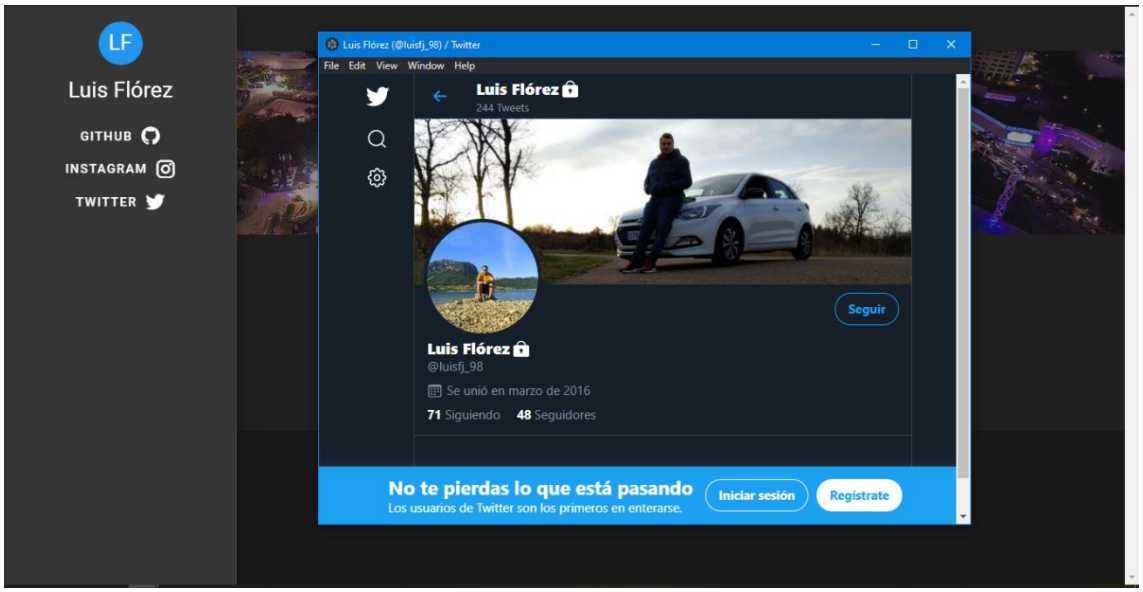
En la siguiente imagen podemos ver el código del combobox y de los botones de aleatorio, recomendar y limpiar.

```
<v-container>
  <v-layout mt-4 column align-center wrap>
    <v-flex xs12>
      <v-autocomplete
        v-model="row"
        :items="items"
        dense
        rounded
        solo
        label="Subgenero"
      ></v-autocomplete>

      <v-btn @click="real" color="orange" outlined dark rounded
        >Aleatorio</v-btn
      >
      <v-btn @click="reco" color="orange" outlined dark rounded
        >Recomendar</v-btn
      >
       <v-btn @click="limpiar" color="orange" outlined dark rounded>Limpiar</v-btn>
    </v-flex>
  </v-layout>
</v-container>
```

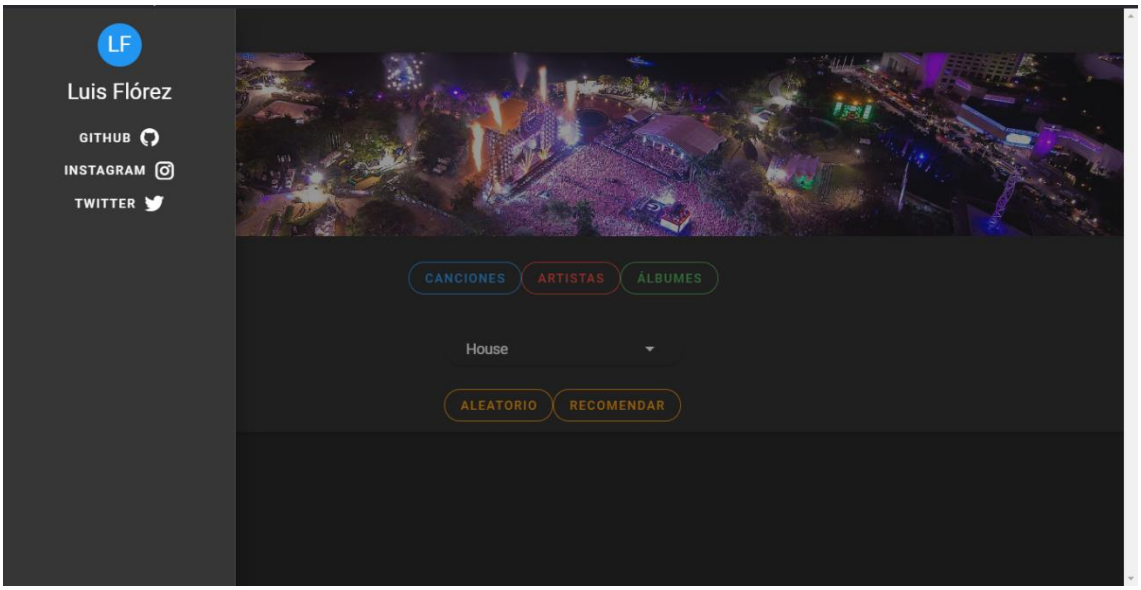
En esta imagen está el código de los botones que muestran las canciones, artistas y álbumes que hay en la base de datos.

```
<v-container>
  <v-layout mt-4 column align-center wrap>
    <v-flex xs12>
      <v-btn @click="can" color="blue" dark outlined rounded
        >Canciones</v-btn
      >

      <v-btn @click="art" color="red" dark outlined rounded
        >Artistas</v-btn
      >

      <v-btn @click="alb" color="green" dark outlined rounded
        >Álbumes</v-btn
      >
    </v-flex>
  </v-layout>
</v-container>
```

Otro componente de la interfaz es el drawer donde muestra el autor de la página web, el GitHub del proyecto y las distintas redes sociales. Este drawer aparece al pulsar el botón con las tres rayas situado arriba a la izquierda de la interfaz.



Este drawer contiene 3 iconos que nos dirigen a cada sitio web.

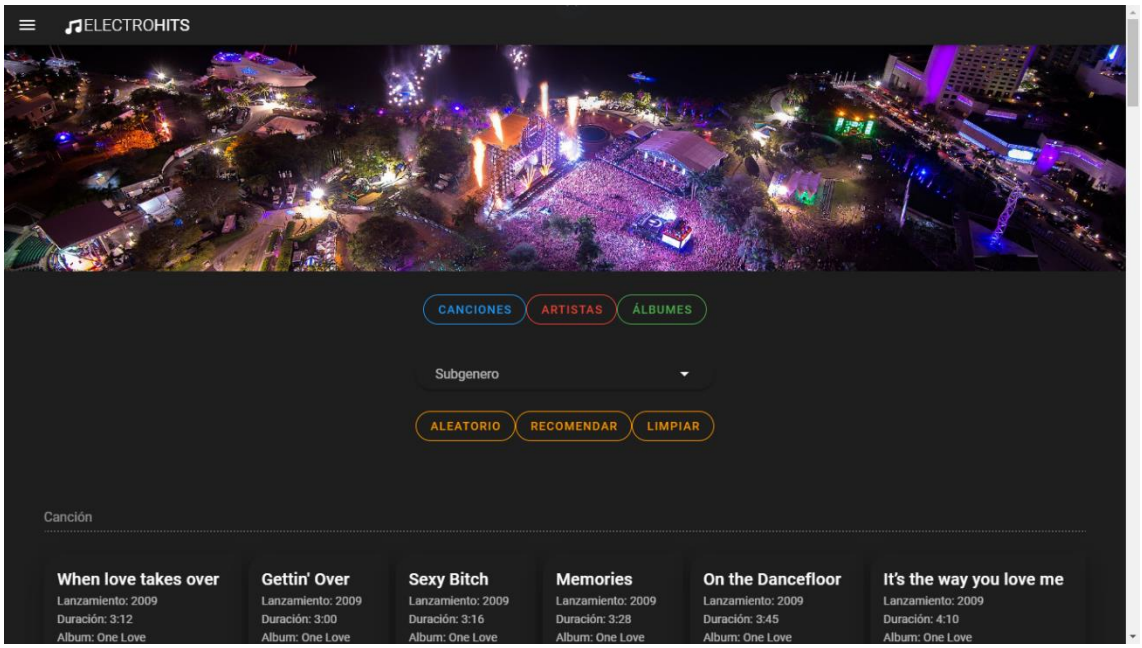


El código de este componente es el siguiente:

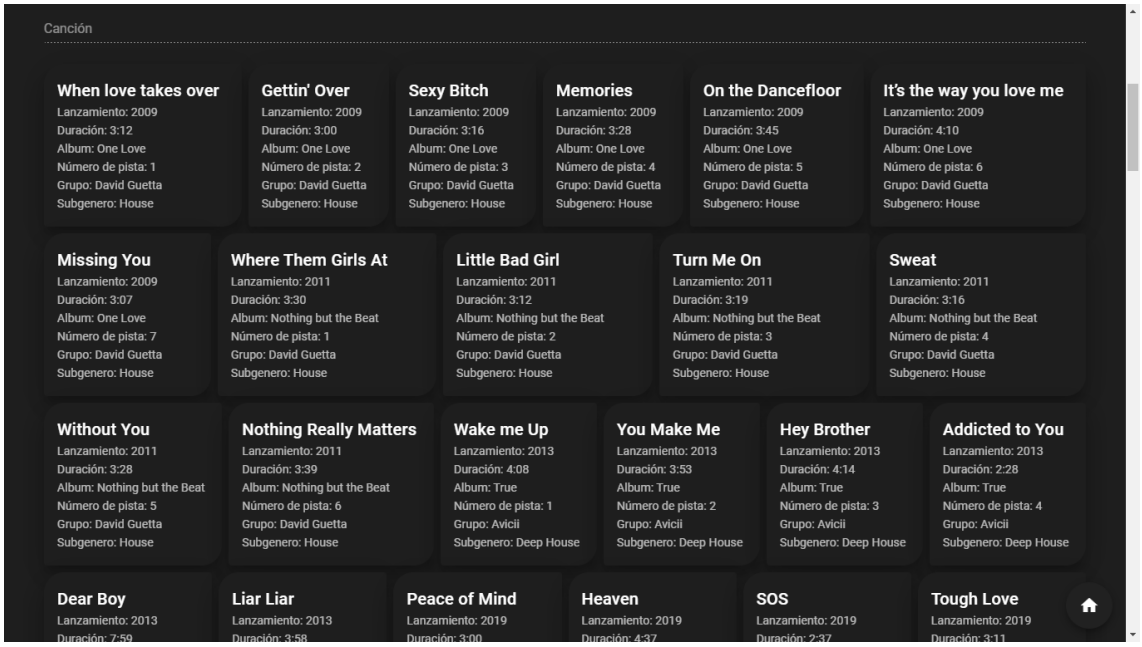
```
<v-navigation-drawer app v-model="drawer" temporary dark>
  <v-layout mt-4 column align-center>
    <v-flex>
      <v-avatar color="blue">
        <span class="white--text headline">LF</span>
      </v-avatar>
    </v-flex>
    <v-flex>
      <p class="white--text mt-3 headline">Luis Flórez</p>
    </v-flex>
    <v-btn href="https://github.com/lflorj00/SIBI" target="_blank" text>
      <span class="mr-2">GitHub</span>
      <v-icon>mdi-github</v-icon>
    </v-btn>
    <v-btn
      href="https://www.instagram.com/luisfj_98/"
      target="_blank"
      text
    >
      <span class="mr-2">Instagram</span>
      <v-icon>mdi-instagram</v-icon>
    </v-btn>
    <v-btn href="https://twitter.com/luisfj_98" target="_blank" text>
      <span class="mr-2">Twitter</span>
      <v-icon>mdi-twitter</v-icon>
    </v-btn>
  </v-layout>
</v-navigation-drawer>
```

En el siguiente apartado voy a explicar cómo funcionan los botones.

Botón Canciones:



Una vez pulsamos el botón nos mostrará todas las canciones que hay en la base de datos con una pequeña información sobre cada canción



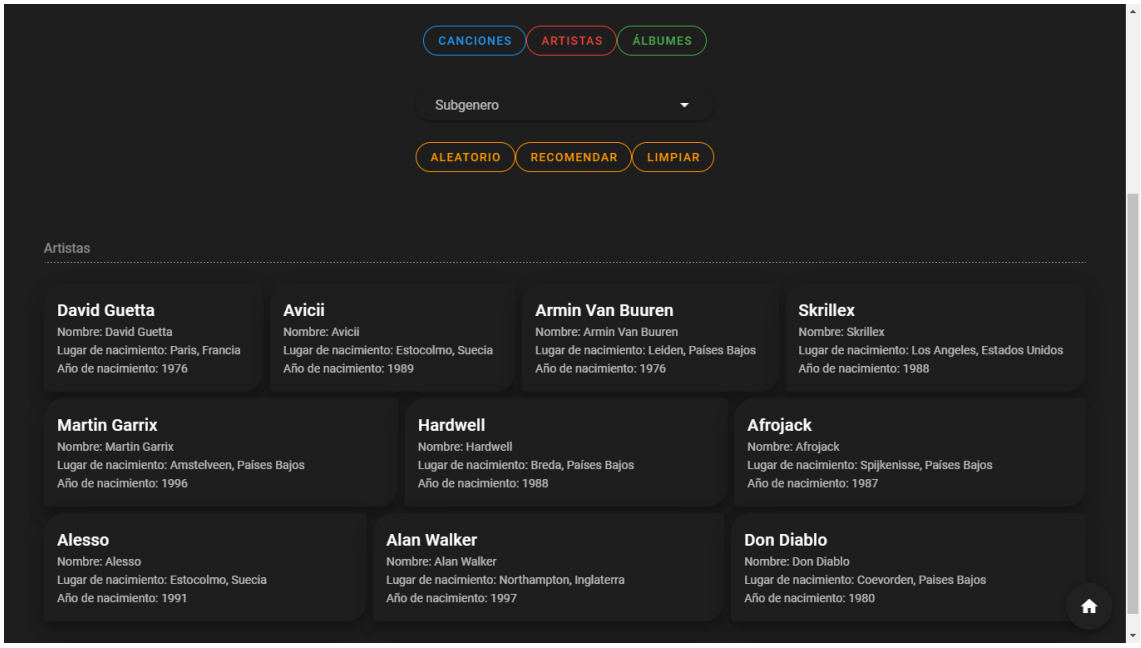
El botón canciones envía una petición desde el frontend esperando respuesta del backend que realiza una consulta de todas las canciones que hay en la base de datos.

```
can() {
  this.buti2 = true;
  this.buti3 = false;
  this.buti4 = false;
  this.buti5 = false;
  this.buti6 = false;
  this.$http.get("http://localhost:8000/can").then((response) => {
    this.array = response.body;
    //alert(response.body);
  });
},
```

El backend recibe la petición y recorre la siguiente función que realiza la consulta en la base de datos:

```
app.get("/can", function (req, res) {
  var query = "MATCH (n:Song) return n";
  var array = [];
  const resultPromise = session.run(query).subscribe({
    onNext: function (record) {
      array.push(record.get(0).properties);
    },
    onCompleted: function () {
      console.log(array);

      res.send(array);

    },
    onError: function (error) {
      console.log(error);
    },
  });
}),
```

Botón Artistas:



Muestra todos los artistas incluidos en la base de datos y además algunos datos como su nombre, fecha de nacimiento y lugar de nacimiento.

El código que muestra las tarjetas con los diferentes artistas de la base de datos es el siguiente:

```
<v-main v-if="this.buti3">
  <v-container grid-list-md text-xs-center fluid pa-12>
      <v-text-field
        label="Artistas"
        v-model="search"
        single-line
        disabled
      ></v-text-field>
        <v-layout row wrap fill-height fill-width justify-center>
      <v-flex v-for="(item, index) in array" v-bind:key="index">
        <v-card elevation="18" shaped dark>
          <v-card-title>{{ item.name }}</v-card-title>
          <v-card-subtitle>
            Nombre: {{ item.name }}
            <br />
            Lugar de nacimiento: {{ item.from }}
            <br />
            Año de nacimiento: {{ item.born }}
          </v-card-subtitle>
        </v-card>
      </v-flex>
    </v-layout>
  </v-container>
</v-main>
```

Cuando pulsamos el botón envía una petición del frontend esperando una respuesta del backend que muestra todos los artistas que hay en la base de datos.

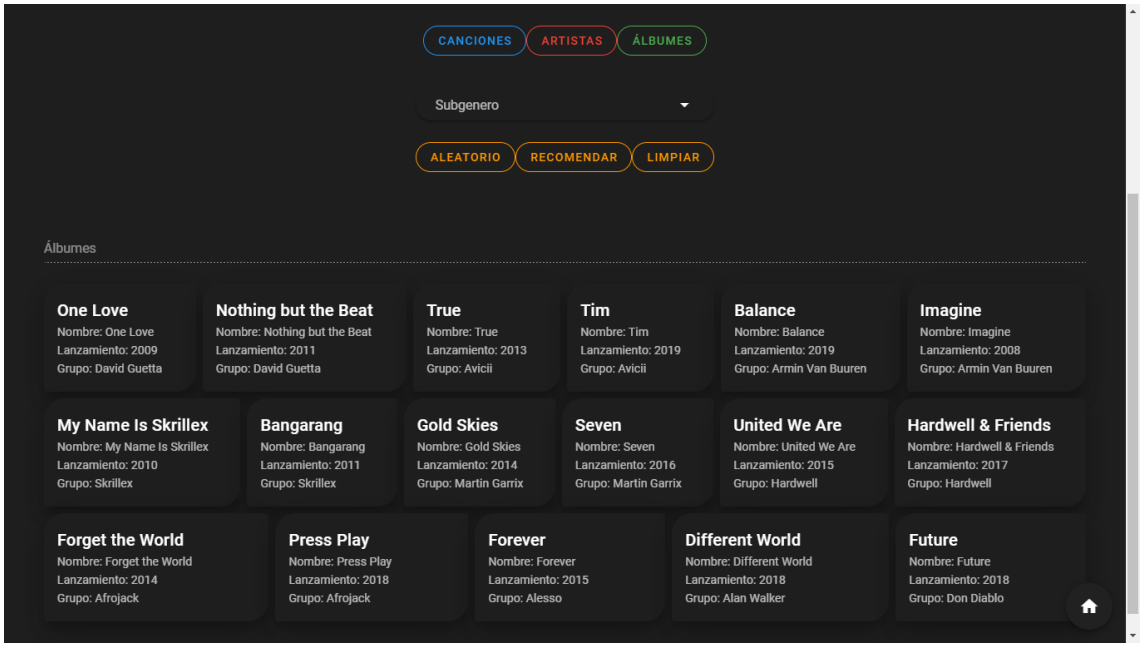
```
art() {
  this.buti2 = false;
  this.buti3 = true;
  this.buti4 = false;
  this.buti5 = false;
  this.buti6 = false;
  this.$http.get("http://localhost:8000/art").then((response) => {
    this.array = response.body;
    //alert(response.body);
  });
},
```

Una vez el backend recibe la petición recorre una función que realiza una consulta a la base de datos.

```
app.get("/art", function (req, res) {
  var query = "match (n) where (:Gender)-[]->(n) return n";
  var array = [];
  const resultPromise = session.run(query).subscribe({
    onNext: function (record) {
      array.push(record.get(0).properties);
    },
    onCompleted: function () {
      console.log(array);

      res.send(array);

    },
    onError: function (error) {
      console.log(error);
    },
  });
});
```

Botón Álbumes:

Muestra todos los álbumes que hay en la base de datos, así como el nombre del álbum, el año de lanzamiento y el artista que lo produjo.



El código que muestra las tarjetas con los álbumes es el siguiente:

```
<v-main v-if="this.buti4">
  <v-container grid-list-md text-xs-center fluid pa-12>
    <v-text-field
      label="Álbumes"
      v-model="search"
      single-line
      disabled
    ></v-text-field>
    <v-layout row wrap fill-height fill-width justify-center>
    <v-flex v-for="(item, index) in array" v-bind:key="index">
      <v-card elevation="18" shaped dark>
        <v-card-title>{{ item.name }}</v-card-title>
        <v-card-subtitle>
          Nombre: {{ item.name }}
          <br />
          Lanzamiento: {{ item.released }}
          <br />
          Grupo: {{ item.group }}
        </v-card-subtitle>
      </v-card>
    </v-flex>
  </v-layout>
</v-container>
</v-main>
</v-card>
```

Una vez pulsamos el botón se envía una petición desde el frontend que espera una repuesta de parte del backend.

```
alb() {
  this.buti2 = false;
  this.buti3 = false;
  this.buti4 = true;
  this.buti5 = false;
  this.buti6 = false;
  this.$http.get("http://localhost:8000/alb").then((response) => {
    this.array = response.body;
    //alert(response.body);
  });
},
```

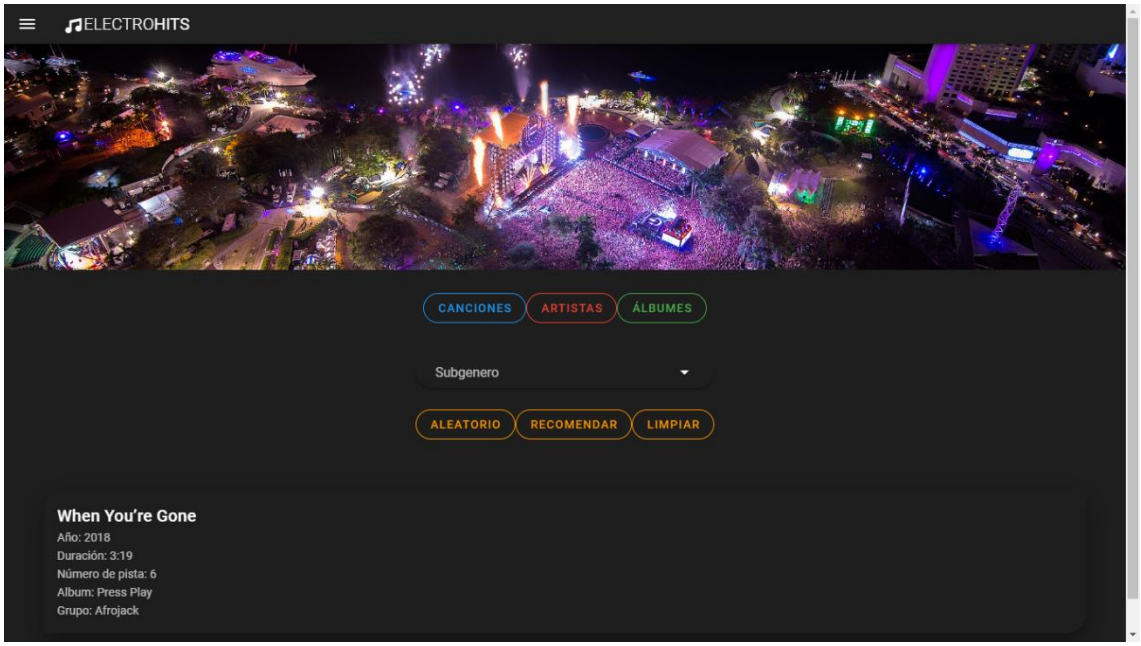
La petición es recibida por el backend y recorre una función que realiza una consulta a la base de datos.

```
app.get("/alb", function (req, res) {
  var query = "MATCH (n:Album) return n";
  var array = [];
  const resultPromise = session.run(query).subscribe({
    onNext: function (record) {
      array.push(record.get(0).properties);
    },
    onCompleted: function () {
      console.log(array);

      res.send(array);

    },
    onError: function (error) {
      console.log(error);
    },
  });
}),
```

4.Algoritmo de la aplicación

Mi aplicación es una página web de recomendación de música en la que tienes dos posibilidades:

- Recomendación aleatoria: te recomienda una canción aleatoria entre todas las canciones que se encuentran en la base de datos.



El código de esta recomendación elige una canción de forma aleatoria entre todas las canciones que hay en la base de datos y la muestra en forma de tarjeta.

```
<v-main v-if="this.buti6">
  <v-container grid-list-md text-xs-center fluid pa-12>
    <v-layout row wrap fill-height fill-width justify-center>
      <v-flex v-for="(item, index) in array" v-bind:key="index">
        <v-card elevation="18" shaped dark min-width="200">
          <v-card-title>{{ item.name }}</v-card-title>
          <v-card-subtitle>
            Año: {{ item.released }}
            <br />
            Duración: {{ item.duration }}
            <br />
            Número de pista: {{ item.trackNumber }}
            <br />
            Album: {{ item.album }}
            <br />
            Grupo: {{ item.group }}
          </v-card-subtitle>
        </v-card>
      </v-flex>
    </v-layout>
  </v-container>
</v-main>
```

Cuando nosotros pulsamos el botón de aleatorio el frontend envía una petición al backend el cual responde con una canción aleatoria de la base de datos.

Esta es la función del frontend que envía la petición y espera una respuesta del servidor.

```
real() {
  //this.tultul = false;
  this.buti2 = false;
  this.buti3 = false;
  this.buti4 = false;
  this.buti5 = false;
  this.buti6 = true;
  this.$http.get("http://localhost:8000/real").then((response) => {
    this.array = response.body;
    //alert(response.body);
  });
},
```

Cuando esa petición llega al backend se recorre esta función la cual realiza una consulta a la base de datos que muestra todas las canciones y elige una de manera aleatoria y la guarda en un array.

```
app.get("/real", function (req, res) {
  var query = "MATCH (n:Song) return n";
  var array = [];

  const resultPromise = session.run(query).subscribe({
    onNext: function (record) {
      array.push(record.get(0).properties);
    },
    onCompleted: function () {
      var randon_int = Math.floor(Math.random() * array.length);
      console.log(array[randon_int]);

      res.send([array[randon_int]]);

    },
    onError: function (error) {
      console.log(error);
    },
  });
}),
```

- Recomendación por subgénero: dentro de la música hay diferentes subgéneros de música electrónica por lo que esta recomendación te mostrara una canción del subgénero que hayas seleccionado previamente.

  Dentro de la base de datos hay canciones con el mismo subgénero que pueden pertenecer a artistas distintos ya que algunos de estos artistas tienen el mismo subgénero.

  Los subgéneros incluidos en la base de datos son los siguientes:

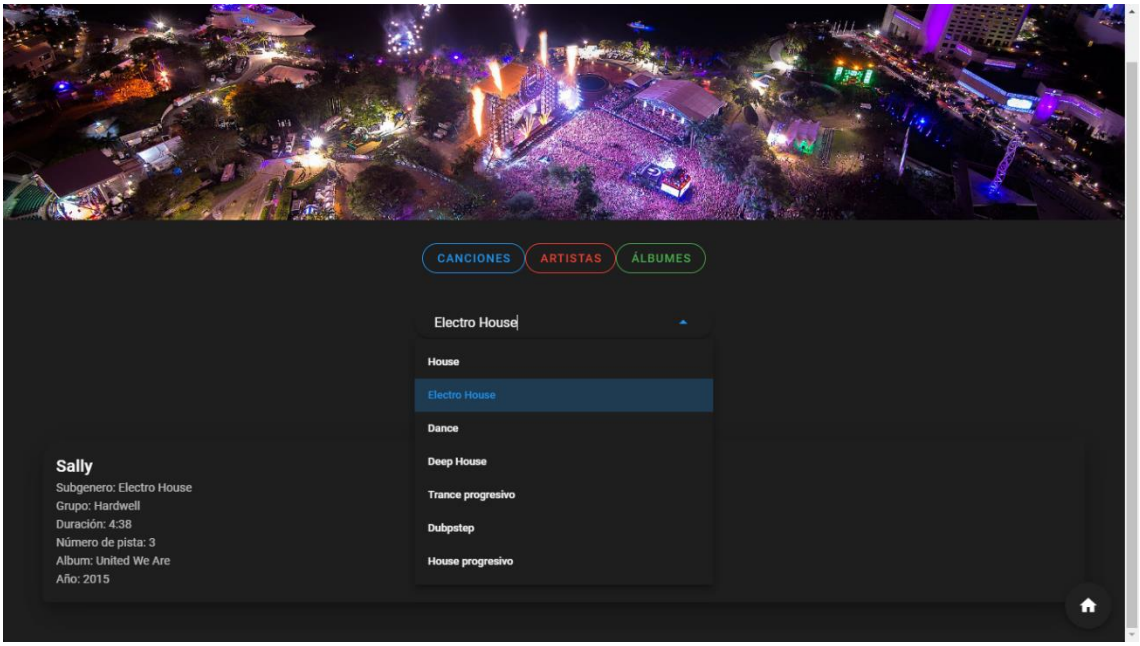
  - House
  - House Progresivo
  - Deep House
  - Electro House
  - Dance
  - Dubpstep
  - Trance Progresivo

  Para que nos muestre el resultado usaremos la siguiente consulta en la base de datos:

  MATCH (n:Song) where n.subgender ='" + subgender + "' return n

  n.subgender = “ + subgender + ” se rellena una vez hayas seleccionado en la interfaz el subgénero y te mostrará una canción de dicho subgénero.

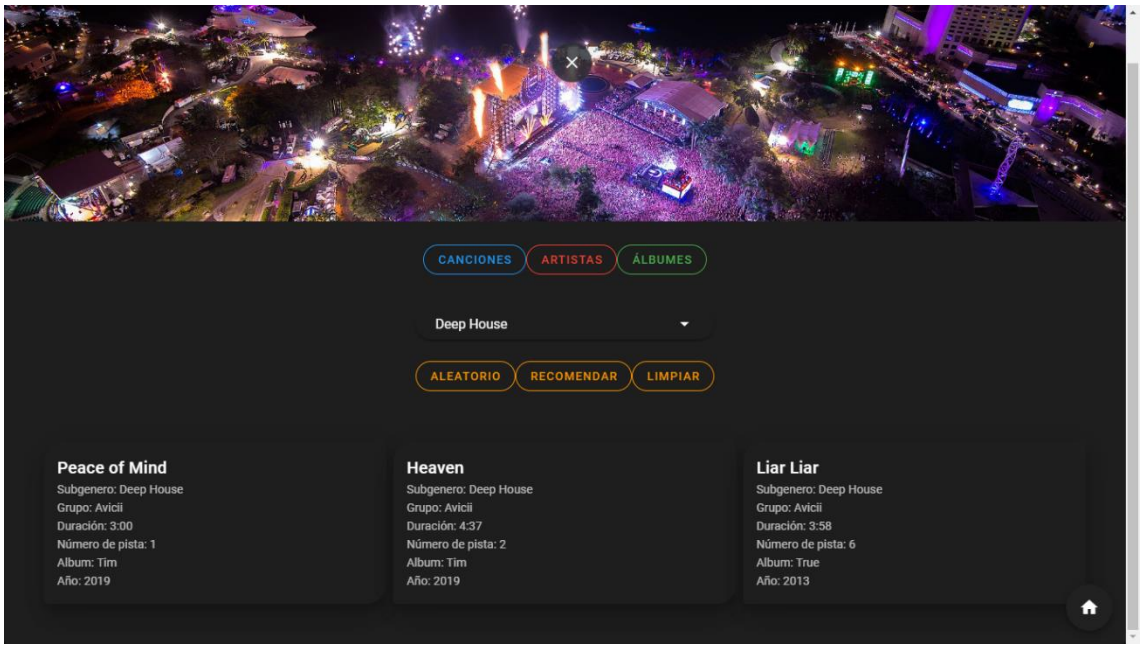
La siguiente imagen nos muestra una lista donde escogemos el subgénero de la canción que queremos que nos muestre.



El código para crear la lista para elegir el subgénero y los botones de aleatorio, recomendar y limpiar es el siguiente:

```
<v-container>
  <v-layout mt-4 column align-center wrap>
    <v-flex xs12>
      <v-autocomplete
        v-model="row"
        :items="items"
        dense
        rounded
        solo
        label="Subgenero"
      ></v-autocomplete>

      <v-btn @click="real" color="orange" outlined dark rounded
        >Aleatorio</v-btn
      >
      <v-btn @click="reco" color="orange" outlined dark rounded
        >Recomendar</v-btn
      >
        <v-btn @click="limpiar" color="orange" outlined dark rounded>Limpiar</v-btn>
    </v-flex>
  </v-layout>
</v-container>
```

Cuando pulsamos el botón de recomendar nos mostrará una canción con el subgénero que hayamos escogido previamente. Si presionamos más veces ese botón nos ira añadiendo las canciones y si queremos limpiar esa lista de canciones presionaremos el botón limpiar



El botón recomendar envía una petición desde el frontend y espera a recibir una respuesta de parte del backend, el cual recorre una función que realiza una consulta a la base de datos con el subgénero escogido y devuelve una canción que guarda en un array.

La petición del frontend es la siguiente:

```
reco() {
  this.buti2 = false;
  this.buti3 = false;
  this.buti4 = false;
  this.buti5 = true;
  this.buti6 = false;
  var subgender = this.row;
  this.$vuetify.goTo(window.innerHeight);
  var data = { subgender: subgender };
  this.$http.post("http://localhost:8000/reco", data).then((response) => {
    this.yarra.push(response.body[0]);
    //alert(response.body);
  });
},
```

El backend recibe la petición y recorre la función que realiza una consulta a la base de datos.

```
app.post("/reco", function (req, res) {
    var subgender = req.body.subgender;
    console.log(subgender);
    var query = "MATCH (n:Song) where n.subgender ='" + subgender + "' return n";

    console.log(query);
    var array = [];
    const resultPromise = session.run(query).subscribe({
      onNext: function (record) {
        array.push(record.get(0).properties);
      },
      onCompleted: function () {
        var randon_int = Math.floor(Math.random() * array.length);
        console.log(array[randon_int]);

        res.send([array[randon_int]]);

      },
      onError: function (error) {
        console.log(error);
      },
    });
  }),
```

5.Analisis de resultados

1º Caso: no conoces la música electrónica y quieres saber que artistas actuales hay y conocer los distintos subgéneros que hay dentro de la música electrónica.

2º Caso: conoces la música electrónica y quieres saber que música han sacado los artistas recientemente o música mas antigua que no conocías.

3º Caso: conoces la música electrónica y quieres que la aplicación te recomiende canciones, ya sea de forma aleatoria o con un subgénero específico.

6.DAFO

Fortalezas	**Amenazas**
- Genero de música especifico. - Aplicación sencilla. - Base de datos propia. - Oportunidad de negocio. - Posibilidad de mejorar la aplicación.	- Posible copia por parte de los competidores.
Debilidades	**Oportunidades**
- Falta de recolección de datos de los usuarios a través de cookies. - Base de datos incompleta. - Poco tiempo de desarrollo	- Ampliar la base de datos. - Mejorar la recopilación de datos de los usuarios. - Crear perfiles de usuario. - Mejorar el algoritmo de recomendación.

7.Lineas de futuro

- Ampliar la base de datos con más artistas y más canciones, así como más información sobre los artistas.
- Añadir buscador de canciones y poderlas añadir a una lista de favoritos de tu perfil.
- Capacidad de crear un usuario en la página web en la que podrás ver tus canciones favoritas, tus artistas favoritos y ver música están escuchando tus amigos.
- Añadir cookies a la página web que recolectaran información sobre que artistas de música electrónica te gustan más y hacer una recomendación personalizada para cada usuario
- Previsualización de la canción ya sea con un plugin de Spotify o redirigiendo a YouTube a través de un enlace.

8.Lecciones aprendidas

En este curso en que he desarrollado una página web de recomendación de música electrónica he aprendido varias cosas:

- Crear bases de datos utilizando neo4j.
- Aprender a realizar consultas en neo4j utilizando el lenguaje Cypher.
- Capacidad de aprender de forma autodidacta.
- Aprender a diseñar páginas web usando Vue.js y Vuetify.
- Aprender un nuevo lenguaje de programación de páginas web que es JavaScript.
- Capacidad de adquirir conocimiento de diferentes fuentes como páginas web o videos de YouTube.

9.Bibliografía

[1] https://neo4j.com/developer/graph-database/

[2] https://es.vuejs.org/v2/guide/

[3] https://developer.mozilla.org/es/docs/Web/JavaScript

[4] https://ifgeekthen.everis.com/es/que-es-node-js-y-primeros-pasos

[5] https://es.wikipedia.org/wiki/Electron_(software)

[6] https://www.luisllamas.es/vuetify-estetica-material-design-para-tus-apps-en-vuejs/