MANUAL TÉCNICO PLATAFORMA ACCESO A CONTENIDOS IPAZUD



ESTUDIANTES
DWAN FELIPE VELOZA PAEZ - 20142020098
IVÁN RENÉ AREVALO VENEGAS - 20142020037

Director

Wilson Díaz Gamba
Psicólogo

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
PROYECTO CURRICULAR INGENIERÍA DE SISTEMAS
BOGOTA D.C
2019

TABLA DE CONTENIDO

BACKEND 2

BASE DE DATOS 12

FRONT END 14

IMPLEMENTACIÓN 36

BACKEND

El backend de nuestro aplicativo web es una API REST y está desarrollado en el lenguaje GO, este se encarga de ingresar a la base de datos (MySQL) y obtener la información necesaria para construir un Json, el cual será retornado mediante una petición http que se realiza por una ruta predefinida

Primero se realizan las importaciones de las librerías que se usarán y se definen un par de funciones para solventar problemas con las cabeceras CORS

Las librerias que se necesitan instalar son las siguientes:
http://github.com/go-sql-driver/mysql
https://github.com/rs/cors
https://github.com/gorilla/mux

```
package main

import (
    _ "github.com/go-sql-driver/mysql" // La librería que nos permite conectar
a MySQL
    "database/sql"                     // Interactuar con bases de datos
    "fmt"                              // Imprimir mensajes y esas cosas
    "github.com/rs/cors"
    "encoding/json"
    "log"
    "net/http"
    "github.com/gorilla/mux"
)

//Funciones para solucionar problema de origen de cabecera

func handler(w http.ResponseWriter, req *http.Request) {
    // ...
    enableCors(&w)
    // ...
}

func enableCors(w *http.ResponseWriter) {
    (*w).Header().Set("Access-Control-Allow-Origin", "*")
```

```
}
```

Ahora se define la funcion que proveerá por la conexion a la base de datos

```
//----------------------------------------SE OBTIENE ACCESO A LA BASE DE DATOS----------------------------------------

func obtenerBaseDeDatos() (db *sql.DB, e error) {
    usuario := "dbipazud" // usuario y contraseña entregados por la red de datos UDNET
    pass := "uoCee3ia.2019"
    host := "tcp(10.20.200.180:3306)" // Dirección del servidor remoto entregado por la red de  datos UDNET y puerto de mysql49

    nombreBaseDeDatos := "pasantia"
    // Debe tener la forma usuario:contraseña@host/nombreBaseDeDatos
    db, err := sql.Open("mysql", fmt.Sprintf("%s:%s@%s/%s", usuario, pass, host, nombreBaseDeDatos))
    if err != nil {
        return nil, err
    }
    return db, nil
}
```

Se define la estructura de los archivos Json para cada tipo de consulta

```
//Estrucutura consulta video
type Video struct {
    Id string `json:"id,omitempty"`
    Tema string `json:"tema,omitempty"`
    Autor string `json:"autor,omitempty"`
    Fecha string `json:"fecha,omitempty"`
    Link string `json:"link,omitempty"`
  }

//Estrucutura consulta audio
type Audio struct {
    Id string `json:"id,omitempty"`
    Tema string `json:"tema,omitempty"`
    Participantes string `json:"participantes,omitempty"`
    Fecha string `json:"fecha,omitempty"`
```

```
    Link string `json:"link,omitempty"`
  }

  //Estrucutura consulta publicidad
type Publicidad struct {
    Id string `json:"id,omitempty"`
    Tema string `json:"tema,omitempty"`
    Tipo string `json:"tipo,omitempty"`
    Participantes string `json:"participantes,omitempty"`
    Fecha string `json:"fecha,omitempty"`
    Link string `json:"link,omitempty"`
  }

    //Estrucutura consulta documentos
type Documento struct {
    Id string `json:"id,omitempty"`
    Tema string `json:"tema,omitempty"`
    Tipo string `json:"tipo,omitempty"`
    Area string `json:"area,omitempty"`
    Autor string `json:"autor,omitempty"`
    Fecha string `json:"fecha,omitempty"`
    Link string `json:"link,omitempty"`
  }
```

Ahora se definen las funciones en las que se define la consulta SQL debido a que estas reciben por parámetros las especificaciones de la búsqueda, estas funciones serán llamadas desde los end points más adelante, su propósito es recibir la información de la base de datos y montar la estructura del Json.

```
  //---------------------------------------SE REALIZA LA CONSULTA PARA
OBTENER VIDEOS ESPECIFICOS-------------------------------
func obtenerVideos(tema string, autor string, fecha string) ([]Video, error) {
    videos := []Video{}
    db, err := obtenerBaseDeDatos()
    if err != nil {
        return nil, err
    }
    defer db.Close()

    //consulta sql
```

```
    filas, err := db.Query("SELECT id, tema, autor, fecha, link FROM videos
where autor LIKE '%"+autor+"%' AND tema LIKE '%"+tema+"%' AND fecha LIKE
'%"+fecha+"%'")

    if err != nil {
        return nil, err
    }
    // Si llegamos aquí, significa que no ocurrió ningún error
    defer filas.Close()

    // Aquí vamos a "mapear" lo que traiga la consulta en el while de más
abajo
    var c Video

    // Recorrer todas las filas, en un "while"
    for filas.Next() {
        err = filas.Scan(&c.Id, &c.Tema, &c.Autor, &c.Fecha , &c.Link)
        // Al escanear puede haber un error
        if err != nil {
            return nil, err
        }

        // Y si no, entonces agregamos lo leído al arreglo

        videos = append(videos, c)
    }
    // Vacío o no, regresamos el arreglo de contactos
    return videos, nil
}

  //---------------------------------------SE REALIZA LA CONSULTA PARA
OBTENER AUDIOS ESPECIFICOS-------------------------------
  func obtenerAudios(tema string, participantes string, fecha string)
([]Audio, error) {
    audios := []Audio{}
    db, err := obtenerBaseDeDatos()
    fmt.Printf(fecha)
    if err != nil {
        return nil, err
    }
```

```
    defer db.Close()

    filas, err := db.Query("SELECT id, tema, participantes, fecha, link FROM
audios where participantes LIKE '%"+participantes+"%' AND tema LIKE
'%"+tema+"%' AND fecha LIKE '%"+fecha+"%'")

    if err != nil {
        return nil, err
    }
    // Si llegamos aquí, significa que no ocurrió ningún error
    defer filas.Close()

    // Aquí vamos a "mapear" lo que traiga la consulta en el while de más
abajo
    var c Audio

    // Recorrer todas las filas, en un "while"
    for filas.Next() {
        err = filas.Scan(&c.Id, &c.Tema, &c.Participantes, &c.Fecha , &c.Link)
        // Al escanear puede haber un error
        if err != nil {
            return nil, err
        }

        // Y si no, entonces agregamos lo leído al arreglo

        audios = append(audios, c)
    }
    // Vacío o no, regresamos el arreglo de contactos
    return audios, nil
}

 //---------------------------------------SE REALIZA LA CONSULTA PARA OBTENER
PUBLICIDAD ESPECIFICOS-------------------------------
 func obtenerPublicidad(tema string, tipo string, participantes string, fecha
string) ([]Publicidad, error) {
    publicidades := []Publicidad{}
    db, err := obtenerBaseDeDatos()
    if err != nil {
        return nil, err
```

```
    }
    defer db.Close()

    filas, err := db.Query("SELECT id, tema, tipo, participantes, fecha, link
FROM publicidad where participantes LIKE '%"+participantes+"%' AND tema LIKE
'%"+tema+"%' AND tipo LIKE '%"+tipo+"%' AND fecha LIKE '%"+fecha+"%'")

    if err != nil {
        return nil, err
    }
    // Si llegamos aquí, significa que no ocurrió ningún error
    defer filas.Close()

    // Aquí vamos a "mapear" lo que traiga la consulta en el while de más
abajo
    var c Publicidad

    // Recorrer todas las filas, en un "while"
    for filas.Next() {
        err = filas.Scan(&c.Id, &c.Tema, &c.Tipo, &c.Participantes, &c.Fecha ,
&c.Link)
        // Al escanear puede haber un error
        if err != nil {
            return nil, err
        }

        // Y si no, entonces agregamos lo leído al arreglo

        publicidades = append(publicidades, c)
    }
    // Vacío o no, regresamos el arreglo de contactos
    return publicidades, nil
}

 //---------------------------------------SE REALIZA LA CONSULTA PARA OBTENER
DOCUMENTOS ESPECIFICOS-------------------------------
 func obtenerDocumento(tema string, tipo string, area string, autor string,
fecha string) ([]Documento, error) {
    documentos := []Documento{}
    db, err := obtenerBaseDeDatos()
```

```
    if err != nil {
        return nil, err
    }
    defer db.Close()

    filas, err := db.Query("SELECT id, tema, tipo, area, autor, fecha, link
FROM documentos where autor LIKE '%"+autor+"%' AND tema LIKE '%"+tema+"%' AND
tipo LIKE '%"+tipo+"%' AND area LIKE '%"+area+"%' AND fecha LIKE
'%"+fecha+"%'")

    if err != nil {
        return nil, err
    }
    // Si llegamos aquí, significa que no ocurrió ningún error
    defer filas.Close()

    // Aquí vamos a "mapear" lo que traiga la consulta en el while de más
abajo
    var c Documento

    // Recorrer todas las filas, en un "while"
    for filas.Next() {
        err = filas.Scan(&c.Id, &c.Tema, &c.Tipo, &c.Area, &c.Autor, &c.Fecha
, &c.Link)
        // Al escanear puede haber un error
        if err != nil {
            return nil, err
        }

        // Y si no, entonces agregamos lo leído al arreglo

        documentos = append(documentos, c)
    }
    // Vacío o no, regresamos el arreglo de contactos
    return documentos, nil
}
```

Una vez realizado todo esto se definen las funciones que serán los end points, es decir las funciones que serán llamadas directamente desde las rutas que vamos a definir para nuestra API REST

```go
// EndPoint Videos
func GetVideoEndpoint(w http.ResponseWriter, req *http.Request){
    w.Header().Set("Content-Type", "application/json")
    params := mux.Vars(req)

  videos, err := obtenerVideos(params["tema"],params["autor"],params["año"])
    if err != nil {
        fmt.Printf("Error obteniendo videos: %v", err)
        return
    }
    var resultado []Video
    for _, video := range videos {
        resultado = append(resultado, Video{Id: video.Id, Tema: video.Tema,
Autor: video.Autor, Fecha: video.Fecha, Link: video.Link})
        fmt.Printf("%v\n", video.Tema)
    }
    fmt.Printf("%v\n", resultado)
    //Se convierte a json y se retorna el resultado
    json.NewEncoder(w).Encode(resultado)
}

// EndPoint Audios
func GetAudioEndpoint(w http.ResponseWriter, req *http.Request){
    w.Header().Set("Content-Type", "application/json")
    params := mux.Vars(req)

  audios, err :=
obtenerAudios(params["tema"],params["participantes"],params["año"])
    if err != nil {
        fmt.Printf("Error obteniendo audios: %v", err)
        return
    }
    var resultado []Audio
    for _, audio := range audios {
        resultado = append(resultado, Audio{Id: audio.Id, Tema: audio.Tema,
Participantes: audio.Participantes, Fecha: audio.Fecha, Link: audio.Link})
        fmt.Printf("%v\n", audio.Tema)
    }
    fmt.Printf("%v\n", resultado)
```

```
    //Se convierte a json y se retorna el resultado
    json.NewEncoder(w).Encode(resultado)
}

// EndPoint publicidad
func GetPublicidadEndpoint(w http.ResponseWriter, req *http.Request){
    w.Header().Set("Content-Type", "application/json")
    params := mux.Vars(req)

  publicidades, err :=
obtenerPublicidad(params["tema"],params["tipo"],params["participantes"],params
["año"])
    if err != nil {
        fmt.Printf("Error obteniendo publicidad: %v", err)
        return
    }
    var resultado []Publicidad
    for _, publicidad := range publicidades {
        resultado = append(resultado, Publicidad{Id: publicidad.Id, Tema:
publicidad.Tema, Tipo: publicidad.Tipo,Participantes:
publicidad.Participantes, Fecha: publicidad.Fecha, Link: publicidad.Link})
        fmt.Printf("%v\n", publicidad.Tema)
    }
    fmt.Printf("%v\n", resultado)
    //Se convierte a json y se retorna el resultado
    json.NewEncoder(w).Encode(resultado)
}

// EndPoint documentos
func GetDocumentosEndpoint(w http.ResponseWriter, req *http.Request){
    w.Header().Set("Content-Type", "application/json")
    params := mux.Vars(req)

  documentos, err :=
obtenerDocumento(params["tema"],params["tipo"],params["area"],params["autor"],
params["año"])
    if err != nil {
        fmt.Printf("Error obteniendo documento: %v", err)
        return
    }
```

```
    var resultado []Documento
    for _, documento := range documentos {
        resultado = append(resultado, Documento{Id: documento.Id, Tema:
documento.Tema, Tipo: documento.Tipo,Area: documento.Area, Autor:
documento.Autor, Fecha: documento.Fecha, Link: documento.Link})
        fmt.Printf("%v\n", documento.Tema)
    }
    fmt.Printf("%v\n", resultado)
    //Se convierte a json y se retorna el resultado
    json.NewEncoder(w).Encode(resultado)
}
```

Ya por último nos queda definir el main , donde también estarán las rutas a las cuales podemos hacer peticiones GET, enviando los respectivos datos de la búsqueda para obtener el json correspondiente

```
func main(){
    router := mux.NewRouter()

    router.HandleFunc("/videos", GetVideosEndpoint).Methods("GET")
    router.HandleFunc("/video/{tema},{autor},{año}",
GetVideoEndpoint).Methods("GET")
    router.HandleFunc("/audio/{tema},{participantes},{año}",
GetAudioEndpoint).Methods("GET")
    router.HandleFunc("/publicidad/{tema},{tipo},{participantes},{año}",
GetPublicidadEndpoint).Methods("GET")
    router.HandleFunc("/documento/{tema},{tipo},{area},{autor},{año}",
GetDocumentosEndpoint).Methods("GET")
    handler := cors.Default().Handler(router)

    //se define el puerto de escucha para nuestro servicio
    log.Fatal(http.ListenAndServe(":3000", handler))
}
```

BASE DE DATOS

La base de datos está compuesta por 4 tablas que no se relacionan entre ellas debido a que las cada tipo de consulta es independiente a las demás, está desarrollada en MySQL y su estructura es la siguiente

estructura tabla videos:

Columna	Tipo
id	int(11)
tema	varchar(100)
autor	varchar(100)
fecha	int(4)
link	varchar(100)

Estructura tabla audios:

Columna	Tipo
id	int(11)
tema	varchar(100)
participantes	varchar(255)
fecha	int(4)
link	varchar(255)

Estructura tabla publicidad:

Columna	Tipo
id	varchar(11)
tema	varchar(255)
tipo	varchar(255)
participantes	varchar(255)
fecha	date
link	varchar(255)

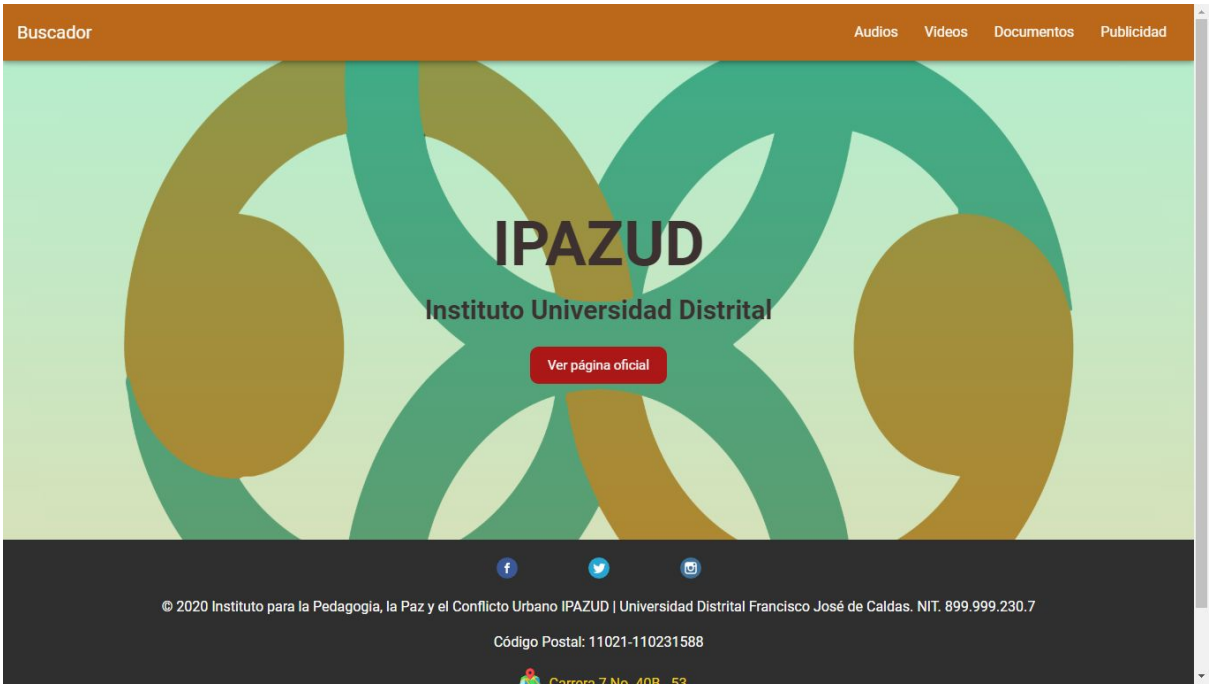
Estructura tabla documentos:

Columna	Tipo
id	int(11)
tema	varchar(100)
tipo	varchar(100)
area	varchar(100)
autor	varchar(100)
fecha	int(4)
link	varchar(100)

FRONT END

El front end se desarrolló con el framework Vue.js, el cual permitió establecer conexión con el servidor de go a través de axios; se desarrollaron 5 pantallas en total de la siguiente manera:

Inicio



Desde la calse App.vue llamamos el componente menu-app que nos permitira navegar entre páginas.

```
<template>
 <!-- APP-Buscador Ipazud
Creada por DWAN FELIPE VELOZA PAEZ e IVAN RENÉ ARÉVALO VENEGAS
Este es el buscador del IPAZUD basado en vue.js y go.-->
<v-app>
    <!-- Llamamos al componente Menu -->
      <menu-app></menu-app>
      <v-content>
          <!-- router-view permite que naveguemos entre pantallas del buscador
-->
          <router-view/>
      </v-content>
```

```
</v-app>
</template>

<style lang="scss">
/* Importamos estilos globales */
@import"sass/globales.scss";
</style>
```

La página principal nos presenta el Home del buscador de la siguiente forma:

```
<template>
<v-layout>
  <!-- Pantalla de inicio -->
     <v-flex class="Presentacion">
        <div class="contenedor">
      <div class="contenedor-titulo">
        <div class="titulo" >
          <h1 class="animated fadeInLeft dura-2">IPAZUD</h1>
          <h2 class="animated fadeInLeft dura-2">Instituto Universidad
Distrital</h2>
          <!-- Enlace rápido con la página principal del IPAZUD -->
          <a class="animated fadeInLeft dura-2"
href="https://ipazud.udistrital.edu.co/" target="_blank" title="Ir a la página
oficial">Ver página oficial</a>
        </div>
      </div>
    </div>
    <!-- Llamamos el componente Footer -->
    <footer-app class="animated fadeInUp dura-1"></footer-app>
     </v-flex>
     </v-layout>
</template>

<style lang="scss" scoped>
/* Importamos clase de estilo sass Principal.scss */
@import "@/sass/Principal.scss";
</style>
```

Como se observa se hace uso del llamado a componentes propios de Vue.js los cuales son menu-app y footer-app

```
<template>
  <div>
    <!-- Uso de vuetify nav y toolbar para desarrollar el menu responsive
    1. drawer es una variable lógica que eprmite desplegar o no el menu en un entorno responsive
    2. La barra de navegación primaria es la que contiene la clase BarraNavegacion.
     -->
    <v-app-bar-nav-icon class="Responsive" @click="drawer=!drawer"></v-app-bar-nav-icon>
    <v-toolbar app color="#BB681A" class="BarraNavegacion">
      <v-toolbar-title>
        <router-link to="./">
          <a class="ItemPrincipal" title="Ir a la página principal">
            Buscador</a>
        </router-link>
      </v-toolbar-title>
      <v-spacer></v-spacer>
      <router-link to="./audios">
        <a class="Secundarios" title="Ir a la categoría Audios">Audios</a>
      </router-link>
      <router-link to="./videos">
        <a class="Secundarios" title="Ir a la categoría Videos">Videos</a>
      </router-link>
      <router-link to="./documentos">
        <a class="Secundarios" title="Ir a la categoría Documentos">Documentos</a>
      </router-link>
      <router-link to="./publicaciones">
        <a class="Secundarios" title="Ir a la categoría Publicidad">Publicidad</a>
      </router-link>
    </v-toolbar>
    <v-navigation-drawer app v-model="drawer" temporary color="#BB681A">
      <v-layout mt-4 ml-2 column align-center>
        <v-flex >
          <v-avatar>
              <router-link to="./">
              <img src="@/assets/LogoIPAZUD.jpg" alt="Logo Ipazud">
```

```
        </router-link>
          </v-avatar>
        </v-flex>
          <v-flex class="white--text mt-2 headline" >
              IPAZUD
          </v-flex>
          <v-divider></v-divider>
          <v-flex>
          <v-list-item class="Items">
            <v-list-item-content>
              <router-link to="./audios">
                <a>Audios</a>
              </router-link>
            </v-list-item-content>
            <v-list-item-content>
              <router-link to="./videos">
                <a>Videos</a>
              </router-link>
            </v-list-item-content>
            <v-list-item-content>
              <router-link to="./documentos">
                <a>Documentos</a>
              </router-link>
            </v-list-item-content>
            <v-list-item-content>
              <router-link to="./publicaciones">
                <a>Publicidad</a>
              </router-link>
            </v-list-item-content>
          </v-list-item>
        </v-flex>
      </v-layout>
    </v-navigation-drawer>
  </div>
</template>

<script>
export default {
  data() {
    return {
```

```
      drawer: false
    };
  }
};
</script>

<style lang="scss" scoped>
/* Estilos de MENU
Display: none permite la interacción entre el responsive con drawer
 */
@media (min-width: 600px) {
    .Responsive {
      display: none;
}
.Secundarios{
    padding: 15px;
    color: white;
}
.Secundarios:hover{
  background: #2E8E83;
  color: #fff;
}
  }
@media (max-width: 600px) {
    .BarraNavegacion {
      display: none;
}
  }

.Items{
    list-style-type: square;
    display: inline;
   padding: 0 1em 0 0;
}
img{
    max-width: 100%;
    width: 100%;
}
.v-application a {
        color: white;
```

```
}

</style>
```

footer-app

```
<template>
   <footer>
   <!-- Contenido del FOOTER -->
        <v-layout class="contenedor" wrapp>
          <!-- Enlazo a través de etiquetas "a" las redes sociales que maneja
el IPAZUD -->
          <v-flex class="redes">
            <a title="Facebook" href="https://www.facebook.com/ipazud/"
target="_blank"><img class="GirarIcon" src='../assets/facebook.png'
alt="imagen de facebook"></a>
            <a title="Twitter" href="https://twitter.com/InstitutoPazUD"
target="_blank"><img class="GirarIcon" src='../assets/twitter.png'
alt="imagen de twitter"></a>
            <a title="Instagram"
href="https://www.instagram.com/institutodepaz_ud/ " target="_blank"><img
class="GirarIcon" src='../assets/instagram.png'  alt="imagen de
instagram"></a>
          </v-flex>
          <!-- Se define la clase parrafo para identificar aspectos de la
Universidad y el IPAZUD -->
          <v-flex class="parrafo">
            <p>© 2020 Instituto para la Pedagogia, la Paz y el Conflicto
Urbano IPAZUD  | Universidad Distrital Francisco José de Caldas. NIT.
899.999.230.7</p>
            <p>Código Postal: 11021-110231588</p>
            <a   title="Ubicación" id="DatosContacto" target="_blank"
href="https://www.google.com.co/maps/place/Cra.+7+%2340b-53,+Bogot%C3%A1/@4.62
80856,-74.0674698,17z/data=!3m1!4b1!4m5!3m4!1s0x8e3f9a287591013f:0x5cce5fbab6b
77b9b!8m2!3d4.6280856!4d-74.0652811?hl=es"><i class="fa fa-map-marker"
aria-hidden="true"></i><img img src='../assets/mapa.png' alt="imagen de
GoogleMaps">Carrera 7 No. 40B - 53 <br>Bogotá D.C. - República de Colombia</a>
            <br>
```

```
            <a  id="DatosContacto"
href="https://www.udistrital.edu.co/directorio" title="Directorio
institucional"><i class="fa fa-phone" aria-hidden="true"></i>Directorio
institucional</a>
          </v-flex>
        </v-layout>
      </footer>
</template>

<style scoped>
/* Estilos del FOOTER */
footer {
  position: absolute;
  bottom: 0;
  justify-content: center;
  background: #2F2F2F;
  padding: 1% 25%;
  text-align: center;
  bottom: 0;
}


footer .contenedor {
  display: flex;
  justify-content: center;
  flex-direction: column;
}

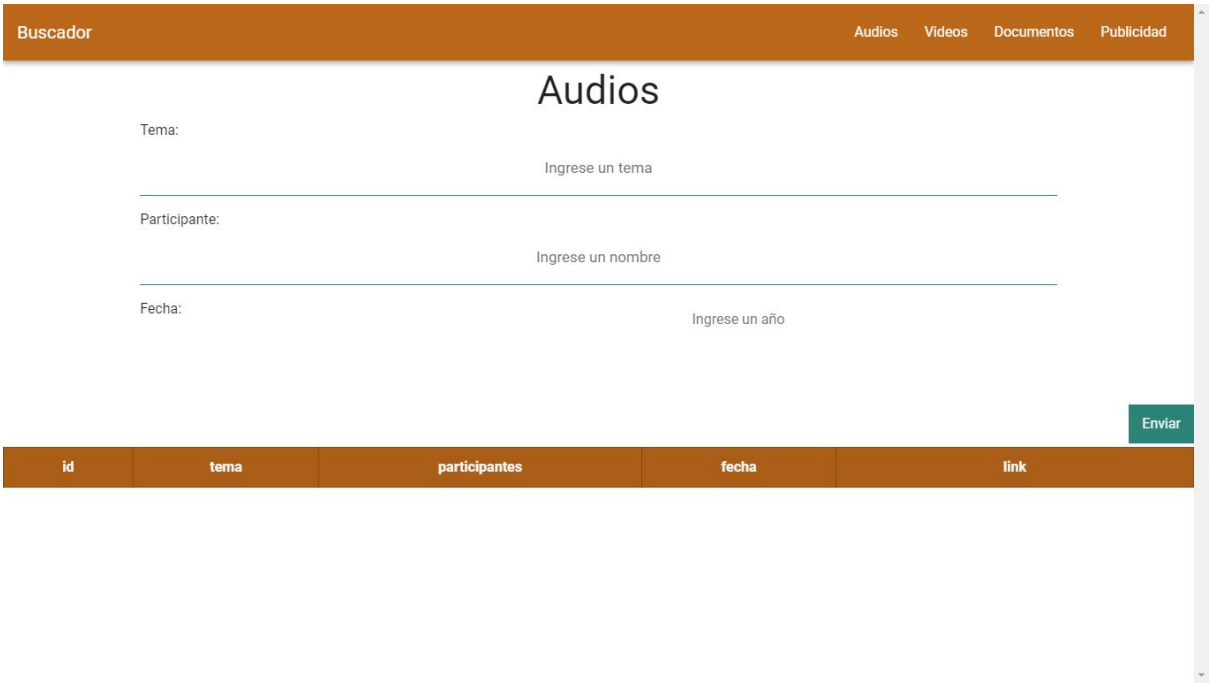
footer .GirarIcon {
  margin: 0 40px;
  transform: rotate(0);
  transition: 0.4s;
}
footer img{
  margin: 0 10px;
}
footer .GirarIcon:hover {
  transform: rotate(360deg);
  transition: 0.4s;
}
```

```
footer p {
  color: #fff;
  margin-top: 1%;
}
#DatosContacto{
  color: rgb(238, 207, 29);
}
@media (max-width: 600px) {
    footer {
      padding: 1% 5%;
}
footer .GirarIcon{
  margin: 0 15px;
}}

</style>
```

Las siguientes páginas corresponden al filtro propuesto en nuestro trabajo de acuerdo a los contenidos más solicitados por la comunidad académica: Audios, Videos, Documentos y Publicidad del IPAZUD.

Audios



Para esta sección el código es el siguiente:

```
<template>
```

```
  <!-- CLASE AUDIOS
Creada por DWAN FELIPE VELOZA PAEZ
Tenga en cuenta la documentación de Vuetify: Grillas, botones y data-tables
Se implementa un buscador dinámico que permite seleccionar dos tipos de
búsqueda: Por filtros y rápida
La búsqueda por filtros requiere de formularios para garantizar que se
incluyan los criterios de búsqueda,
mientras que la búsqueda rápida trae todo el contenido de la tabla en la Base
de Datos y realiza filtros
  mediante search.-->
  <v-container class="animated fadeInUp dura-1">
    <H3>AUDIOS</H3>
    <!-- Botones de inicio -->
    <v-layout v-if="VerBotones" wrap mt-5 xs12 justify-center>
      <div class="Seleccion">
        <v-btn
          title="Búsqueda por Filtros"
          class="mt-5"
          color="success"
          @click="elegirFiltros"
        >Búsqueda Por Filtros</v-btn>
        <v-btn
          color="success"
          class="mt-5"
          @click="elegirBusquedaRapida"
          title="Búsqueda Rápida"
        >Búsqueda Rápida</v-btn>
      </div>
    </v-layout>
    <!-- Invocamos el modal para control de errores -->
    <modal-app v-if="showModal" @close="showModal=!showModal">
      <h3 slot="header">Audio no encontrado</h3>
    </modal-app>
    <!-- Formulario para la búsqueda por filtros -->
    <v-layout v-if="VerFiltros" class="animated fadeIn dura-1" wrap>
      <v-flex xs12>
        <v-form ref="form" v-model="valid" lazy-validation>
          <v-text-field v-model="form.tema" :rules="temaRules" label="Tema"
required></v-text-field>
```

```
          <v-text-field v-model="form.autor" :rules="autorRules"
label="Participantes" required></v-text-field>

          <v-text-field v-model="form.fecha" :rules="fechaRules" label="Fecha"
required></v-text-field>

          <btn :disabled="!valid" class="mr-4"
@click="creartablaAudios">Enviar</btn>

          <btn class="mr-4" @click="reset">Reestablecer Búsqueda</btn>
        </v-form>
      </v-flex>
      <v-flex mt-4>
        <v-card-title v-if="showSearch">
          <v-text-field
            v-model="search"
            append-icon="mdi-magnify"
            label="Search"
            single-line
            hide-details
          ></v-text-field>
        </v-card-title>
        <v-data-table v-if="showSearch" :headers="headers" :items="misAudios"
:search="search">
          <template v-slot:item.link="{ item }">
            <iframe
              width="350"
              height="250"
              scrolling="no"
              frameborder="no"
              allow="autoplay"
              :src="item.link"
            ></iframe>
          </template>
        </v-data-table>
      </v-flex>
    </v-layout>
    <!-- Búsqueda Rápida -->
    <v-flex v-if="Fast" mt-4 class="animated fadeIn dura-1">
      <v-card-title>
```

```
        <v-text-field
          v-model="search"
          append-icon="mdi-magnify"
          label="Search"
          single-line
          hide-details
        ></v-text-field>
      </v-card-title>
      <v-data-table :headers="headers" :items="misAudios" :search="search">
        <template v-slot:item.link="{ item }">
          <iframe
            width="350"
            height="250"
            scrolling="no"
            frameborder="no"
            allow="autoplay"
            :src="item.link"
          ></iframe>
        </template>
      </v-data-table>
    </v-flex>
  </v-container>
</template>
```

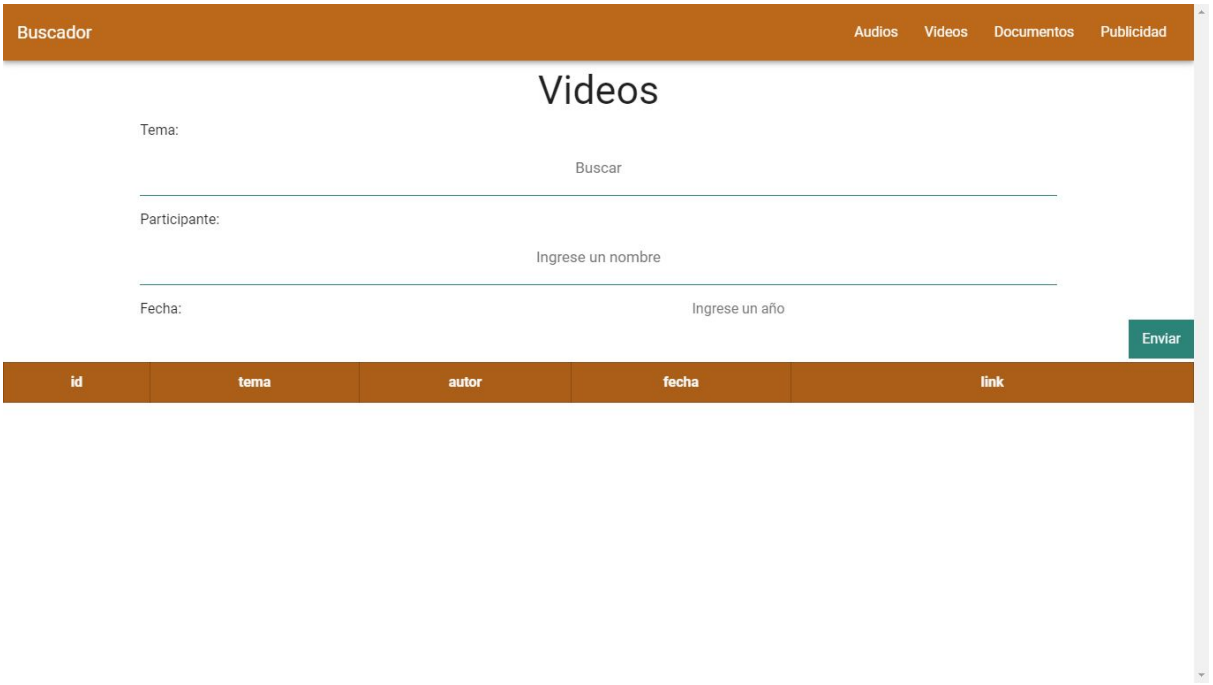
El script que contiene la función para realizar la consulta por filtros y consulta rápida es el siguiente:

```
<script>
export default {
  /* En esta sección declaramos variables que permiten cambiar de tipo de
búsqueda
  también se incluyen los v-model necesarios para generar la búsqueda más
adelante. */
  data() {
    return {
      VerBotones: true,
      Fast: false,
      VerFiltros: false,
      showSearch: false,
      headers: [
```

```
        {
          text: "Id",
          align: "start",
          sortable: false,
          value: "id"
        },
        { text: "Tema", value: "tema" },
        { text: "Participantes", value: "participantes" },
        { text: "Fecha", value: "fecha" },
        { text: "Link", value: "link" }
      ],
      search: "",
      valid: true,
      showModal: false,
      misAudios: [],
      temaRules: [v => !!v || "Tema es requerido"],
      autorRules: [v => !!v || "Autor es requerido"],
      fechaRules: [v => !!v || "Fecha es requerida"],
      form: {
        tema: "",
        autor: "",
        fecha: ""
      },
      show: true
    };
  },
  methods: {
    /* Los métodos creartablaAudios y  elegirBusquedaRapida requieren de Axios
para generar búsquedas por método GET
    Una vez se despliegue el servidor en la red de datos con una ip pública se
debe copiar y reemplazar por el localhost*/
    creartablaAudios() {
      this.axios
        .get(
          "http://localhost:3000/audio/" +
            this.form.tema +
            "," +
            this.form.autor +
            "," +
            this.form.fecha
```

```
        )
        .then(res => {
          this.misAudios = res.data;
          this.showSearch = true;
          if( this.misAudios==null){
            this.showModal = true;
            this.misAudios=[];
          }
        })
        .catch(e => {
          this.showModal = true;
          console(e.response);
        });
    },
    /* Limpiamos campos del formulario */
    reset() {
      this.$refs.form.reset();
    },
    /* Método para solicitar una búsqueda por filtros */
    elegirFiltros() {
      this.VerBotones = false;
      this.VerFiltros = true;
    },
    elegirBusquedaRapida() {
      this.VerBotones = false;
      this.Fast = true;
      this.axios
        .get("http://localhost:3000/audios")
        .then(res => {
          this.misAudios = res.data;
        })
        .catch(e => {
          this.showModal = true;
          console(e.response);
        });
    }
  }
};
</script>
```

Videos



Para esta sección el código es similar al de los audios y es el siguiente:

```
<template>
<!-- CLASE DOCUMENTOS
Creada por DWAN FELIPE VELOZA PAEZ
Tenga en cuenta la documentación de Vuetify: Grillas, botones y data-tables
Se implementa un buscador dinámico que permite seleccionar dos tipos de
búsqueda: Por filtros y rápida
La búsqueda por filtros requiere de formularios para garantizar que se
incluyan los criterios de búsqueda,
mientras que la búsqueda rápida trae todo el contenido de la tabla en la Base
de Datos y realiza filtros
  mediante search.-->
  <v-container class="animated fadeInUp dura-1">
    <H3>VIDEOS</H3>
        <!-- Botones de inicio -->
    <v-layout v-if="VerBotones" wrap mt-5 xs12 justify-center>
      <div class="Seleccion">
        <v-btn
          title="Búsqueda por Filtros"
          class="mt-5"
          color="success"
```

```
          @click="elegirFiltros"
        >Búsqueda Por Filtros</v-btn>
        <v-btn
          color="success"
          class="mt-5"
          @click="elegirBusquedaRapida"
          title="Búsqueda Rápida"
        >Búsqueda Rápida</v-btn>
      </div>
    </v-layout>
        <!-- Invocamos el modal para control de errores -->
    <modal-app v-if="showModal" @close="showModal=!showModal">
      <h3 slot="header">Video no encontrado</h3>
    </modal-app>
        <!-- Formulario para la búsqueda por filtros -->
    <v-layout v-if="VerFiltros" wrap class="animated fadeIn dura-1">
      <v-flex xs12>
        <v-form ref="form" v-model="valid" lazy-validation>
          <v-text-field v-model="form.tema" :rules="temaRules" label="Tema"
required></v-text-field>

          <v-text-field v-model="form.autor" :rules="autorRules"
label="Participantes" required></v-text-field>

          <v-text-field v-model="form.fecha" :rules="fechaRules" label="Fecha"
required></v-text-field>

          <btn class="mr-4" @click="creartablaVideos">Enviar</btn>

          <btn class="mr-4" @click="reset">Reestablecer Búsqueda</btn>
        </v-form>
      </v-flex>
      <v-flex mt-4>
        <v-card-title v-if="showSearch">
          <v-text-field
            v-model="search"
            append-icon="mdi-magnify"
            label="Search"
            single-line
            hide-details
```

```
        ></v-text-field>
      </v-card-title>
      <v-data-table :headers="headers" :items="misVideos" :search="search">
        <template v-slot:item.link="{ item }">
          <iframe
            width="550"
            height="450"
            scrolling="no"
            frameborder="no"
            allow="autoplay"
            :src="item.link"
          ></iframe>
        </template>
      </v-data-table>
    </v-flex>
  </v-layout>
      <!-- Búsqueda Rápida -->
  <v-flex v-if="Fast" mt-4 class="animated fadeIn dura-1">
    <v-card-title>
      <v-text-field
        v-model="search"
        append-icon="mdi-magnify"
        label="Search"
        single-line
        hide-details
      ></v-text-field>
    </v-card-title>
    <v-data-table :headers="headers" :items="misVideos" :search="search">
      <template v-slot:item.link="{ item }">
        <iframe
          width="550"
          height="450"
          scrolling="no"
          frameborder="no"
          allow="autoplay"
          :src="item.link"
        ></iframe>
      </template>
    </v-data-table>
  </v-flex>
```

```
  </v-container>
</template>
```

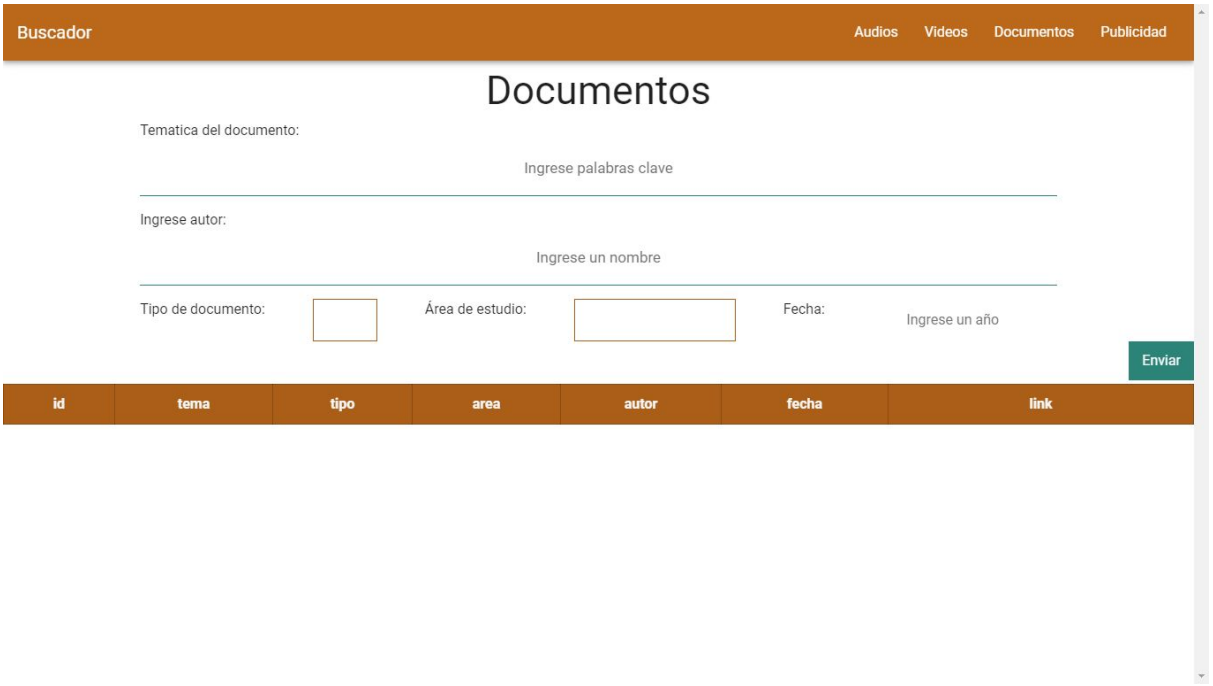
El script que contiene la función creartablaVideos es el siguiente:

```
<script>
export default {
  /* En esta sección declaramos variables que permiten cambiar de tipo de
búsqueda
  tambien se incluyen lo v-model necesarios para generar la búsqueda más
adelante. */
  data() {
    return {
      VerBotones: true,
      Fast: false,
      VerFiltros: false,
      showSearch: false,
      headers: [
        {
          text: "Id",
          align: "start",
          sortable: false,
          value: "id"
        },
        { text: "Tema", value: "tema" },
        { text: "Participantes", value: "autor" },
        { text: "Fecha", value: "fecha" },
        { text: "Link", value: "link" }
      ],
      search: "",
      valid: true,
      misVideos: [],
      showModal: false,
      temaRules: [v => !!v || "Tema es requerido"],
      autorRules: [v => !!v || "Autor es requerido"],
      fechaRules: [v => !!v || "Fecha es requerida"],
      form: {
        tema: "",
```

```
        autor: "",
        fecha: ""
      },
      show: true
    };
  },
  methods: {
    /* Los métodos creartablaAudios y  elegirBusquedaRapida requieren de Axios
para generar búsquedas por método GET
    Una vez se despliegue el servidor en la red de datos con una ip pública se
debe copiar y reemplazar por el localhost*/
    creartablaVideos() {
      this.axios
        .get(
          "http://localhost:3000/video/" +
            this.form.tema +
            "," +
            this.form.autor +
            "," +
            this.form.fecha
        )
        .then(res => {
          this.misVideos = res.data;
          this.showSearch = true;
          if( this.misVideos==null){
            this.showModal = true;
            this.misVideos=[];
          }
        })
        .catch(e => {
          this.showModal = true;
          console(e.response);
        });
    },
    /* Limpiamos campos del formulario */
    reset() {
      this.$refs.form.reset();
    },
        /* Método para solicitar una búsqueda por filtros */
    elegirFiltros() {
```

```
      this.VerBotones = false;
      this.VerFiltros = true;
    },
    elegirBusquedaRapida() {
      this.VerBotones = false;
      this.Fast = true;
      this.axios
        .get("http://localhost:3000/videos")
        .then(res => {
          this.misVideos = res.data;
        })
        .catch(e => {
          this.showModal = true;
          console(e.response);
        });
    }
  }
};
</script>
```

Documentos



El codigo de esta sección es el siguiente:

```
 <template>
  <!-- CLASE DOCUMENTOS
Creada por DWAN FELIPE VELOZA PAEZ
```

```
Tenga en cuenta la documentación de Vuetify: Grillas, botones y data-tables
Se implementa un buscador dinámico que permite seleccionar dos tipos de
búsqueda: Por filtros y rápida
La búsqueda por filtros requiere de formularios para garantizar que se
incluyan los criterios de búsqueda,
mientras que la búsqueda rápida trae todo el contenido de la tabla en la Base
de Datos y realiza filtros
  mediante search.-->
  <v-container class="animated fadeInUp dura-1">
    <h3>DOCUMENTOS</h3>
    <!-- Botones de inicio -->
    <v-layout v-if="VerBotones" wrap mt-5 xs12 justify-center>
      <div class="Seleccion">
        <v-btn
          title="Búsqueda por Filtros"
          class="mt-5"
          color="success"
          @click="elegirFiltros"
        >Búsqueda Por Filtros</v-btn>
        <v-btn
          color="success"
          class="mt-5"
          @click="elegirBusquedaRapida"
          title="Búsqueda Rápida"
        >Búsqueda Rápida</v-btn>
      </div>
    </v-layout>
    <!-- Invocamos el modal para control de errores -->
    <modal-app v-if="showModal" @close="showModal=!showModal">
      <h3 slot="header">Documento no encontrado</h3>
    </modal-app>
    <!-- Formulario para la búsqueda por filtros -->
    <v-layout v-if="VerFiltros" wrap class="animated fadeIn dura-1">
      <v-flex xs12>
        <v-form ref="form" v-model="valid" lazy-validation>
          <v-text-field v-model="form.tema" :rules="temaRules" label="Tema"
required></v-text-field>

          <v-text-field v-model="form.autor" :rules="autorRules" label="Autor"
required></v-text-field>
```

```
          <v-flex xs6 md3>
            <v-select
              v-model="form.tipo"
              :items="form.select"
              :rules="[v => !!v || 'Item es requerido']"
              label="Tipo de Documento"
              required
            ></v-select>
            <v-select
              v-model="form.area"
              :items="form.areas"
              :rules="[v => !!v || 'Item es requerido']"
              label="Area"
              required
            ></v-select>
          </v-flex>

          <v-text-field v-model="form.fecha" :rules="fechaRules" label="Fecha"
required></v-text-field>

          <btn :disabled="!valid" class="mr-4"
@click="creartablaDocumentos">Enviar</btn>

          <btn class="mr-4" @click="reset">Reestablecer Búsqueda</btn>
        </v-form>
      </v-flex>
      <v-flex mt-4>
        <v-card-title v-if="showSearch">
          <v-text-field
            v-model="search"
            append-icon="mdi-magnify"
            label="Search"
            single-line
            hide-details
          ></v-text-field>
        </v-card-title>
        <v-data-table :headers="headers" :items="misDocumentos"
:search="search">
          <template v-slot:item.link="{ item }">
```

```
            <iframe :src="item.link" scrolling="auto" width="550"
height="450"></iframe>
          </template>
        </v-data-table>
      </v-flex>
    </v-layout>
    <!-- Búsqueda Rápida -->
    <v-flex v-if="Fast" mt-4 class="animated fadeIn dura-1">
      <v-card-title>
        <v-text-field
          v-model="search"
          append-icon="mdi-magnify"
          label="Search"
          single-line
          hide-details
        ></v-text-field>
      </v-card-title>
      <v-data-table :headers="headers" :items="misDocumentos"
:search="search">
        <template v-slot:item.link="{ item }">
          <iframe :src="item.link" scrolling="auto" width="550"
height="450"></iframe>
        </template>
      </v-data-table>
    </v-flex>
  </v-container>
</template>
```

El script que contiene la función creartablaDocumentos es el siguiente:

```
<script>
export default {
  /* En esta sección declaramos variables que permiten cambiar de tipo de
búsqueda
  tambien se incluyen lo v-model necesarios para generar la búsqueda más
adelante. */
  data() {
    return {
```

```
    VerBotones: true,
    Fast: false,
    VerFiltros: false,
    showSearch: false,
    headers: [
      {
        text: "Id",
        align: "start",
        sortable: false,
        value: "id"
      },
      { text: "Tema", value: "tema" },
      { text: "Tipo", value: "tipo" },
      { text: "Area", value: "area" },
      { text: "Autor", value: "autor" },
      { text: "Fecha", value: "fecha" },
      { text: "Link", value: "link" }
    ],
    search: "",
    valid: true,
    showModal: false,
    misDocumentos: [],
    temaRules: [v => !!v || "Tema es requerido"],
    autorRules: [v => !!v || "Autor es requerido"],
    fechaRules: [v => !!v || "Fecha es requerida"],
    form: {
      tema: "",
      tipo: "",
      select: [
        "Libro",
        "Revista",
        "Artículo",
        "Tesis",
        "Informe",
        "Entrevista",
        "Registro"
      ],
      area: "",
      areas: [
        "Democracia y Ciudadanía",
```

```
          "Memoria y Conflicto",
          "Territorio y Desarraigo"
        ],
        autor: "",
        fecha: ""
      },
      show: true
    };
  },
  methods: {
    /* Los métodos creartablaAudios y  elegirBusquedaRapida requieren de Axios
para generar búsquedas por método GET
    Una vez se despliegue el servidor en la red de datos con una ip pública se
debe copiar y reemplazar por el localhost*/
    creartablaDocumentos() {
      this.axios
        .get(
          "http://localhost:3000/documento/" +
            this.form.tema +
            "," +
            this.form.tipo +
            "," +
            this.form.area +
            "," +
            this.form.autor +
            "," +
            this.form.fecha
        )
        .then(res => {
          this.misDocumentos = res.data;
          this.showSearch = true;
           if( this.misDocumentos==null){
            this.showModal = true;
            this.misDocumentos=[];
          }
        })
        .catch(e => {
          this.showModal = true;
          console(e.response);
        });
```

```
    },
    /* Limpiamos campos del formulario */
    reset() {
      this.$refs.form.reset();
    },
    /* Método para solicitar una búsqueda por filtros */
    elegirFiltros() {
      this.VerBotones = false;
      this.VerFiltros = true;
    },
    elegirBusquedaRapida() {
      this.VerBotones = false;
      this.Fast = true;
      this.axios
        .get("http://localhost:3000/documentos")
        .then(res => {
          this.misDocumentos = res.data;
        })
        .catch(e => {
          this.showModal = true;
          console(e.response);
        });
    }
  }
};
</script>
```

Publicaciones



El código de esta sección es el siguiente:

```
<template>
  <!-- CLASE PUBLICACIONES
Creada por DWAN FELIPE VELOZA PAEZ
Tenga en cuenta la documentación de Vuetify: Grillas, botones y data-tables
Se implementa un buscador dinámico que permite seleccionar dos tipos de
búsqueda: Por filtros y rápida
La búsqueda por filtros requiere de formularios para garantizar que se
incluyan los criterios de búsqueda,
mientras que la búsqueda rápida trae todo el contenido de la tabla en la Base
de Datos y realiza filtros
  mediante search.-->
  <v-container class="animated fadeInUp dura-1">
    <h3>PUBLICACIONES</h3>
    <!-- Botones de inicio -->
    <v-layout v-if="VerBotones" wrap mt-5 xs12 justify-center>
      <div class="Seleccion">
        <v-btn
          title="Búsqueda por Filtros"
          class="mt-5"
          color="success"
          @click="elegirFiltros"
        >Búsqueda Por Filtros</v-btn>
        <v-btn
          color="success"
          class="mt-5"
          @click="elegirBusquedaRapida"
          title="Búsqueda Rápida"
        >Búsqueda Rápida</v-btn>
```

```
      </div>
    </v-layout>
    <!-- Invocamos el modal para control de errores -->
    <modal-app v-if="showModal" @close="showModal=!showModal">
      <h3 slot="header">Publicación no encontrada</h3>
    </modal-app>
    <!-- Formulario para la búsqueda por filtros -->
    <v-layout v-if="VerFiltros" wrap class="animated fadeIn dura-1">
      <v-flex xs12>
        <v-form ref="form" v-model="valid" lazy-validation>
          <v-text-field v-model="form.tema" :rules="temaRules" label="Tema"
required></v-text-field>

          <v-text-field
            v-model="form.Participante"
            :rules="autorRules"
            label="Participante"
            required
          ></v-text-field>
          <v-flex xs6 md3>
            <v-select
              v-model="form.tipo"
              :items="form.select"
              :rules="[v => !!v || 'Item es requerido']"
              label="Tipo de Publicidad"
              required
            ></v-select>
          </v-flex>

          <v-text-field v-model="form.fecha" :rules="fechaRules" label="Fecha"
required></v-text-field>

          <btn :disabled="!valid" class="mr-4"
@click="creartablaPublicidad">Enviar</btn>

          <btn class="mr-4" @click="reset">Reestablecer Búsqueda</btn>
        </v-form>
      </v-flex>
      <v-flex mt-4>
        <v-card-title v-if="showSearch">
```

```
          <v-text-field
            v-model="search"
            append-icon="mdi-magnify"
            label="Search"
            single-line
            hide-details
          ></v-text-field>
        </v-card-title>
        <v-data-table :headers="headers" :items="misPublicaciones"
:search="search">
          <template v-slot:item.link="{ item }">
            <iframe :src="item.link" scrolling="auto" width="550"
height="450"></iframe>
          </template>
        </v-data-table>
      </v-flex>
    </v-layout>
    <!-- Búsqueda Rápida -->

    <v-flex v-if="Fast" mt-4 class="animated fadeIn dura-1">
      <v-card-title>
        <v-text-field
          v-model="search"
          append-icon="mdi-magnify"
          label="Search"
          single-line
          hide-details
        ></v-text-field>
      </v-card-title>
      <v-data-table :headers="headers" :items="misPublicaciones"
:search="search">
        <template v-slot:item.link="{ item }">
          <iframe :src="item.link" scrolling="auto" width="550"
height="450"></iframe>
        </template>
      </v-data-table>
    </v-flex>
  </v-container>
</template>
```

El script que contiene la función creartablaPublicidad es el siguiente:

```
  <script>
export default {
  /* En esta sección declaramos variables que permiten cambiar de tipo de
búsqueda
  tambien se incluyen lo v-model necesarios para generar la búsqueda más
adelante. */
  data() {
    return {
      VerBotones: true,
      Fast: false,
      VerFiltros: false,
      showSearch: false,
      headers: [
        {
          text: "Id",
          align: "start",
          sortable: false,
          value: "id"
        },
        { text: "Tema", value: "tema" },
        { text: "Tipo", value: "tipo" },
        { text: "Participante", value: "participantes" },
        { text: "Fecha", value: "fecha" },
        { text: "Link", value: "link" }
      ],
      search: "",
      valid: true,
      showModal: false,
      temaRules: [v => !!v || "Tema es requerido"],
      autorRules: [v => !!v || "Autor es requerido"],
      fechaRules: [v => !!v || "Fecha es requerida"],
      misPublicaciones: [],
      form: {
        tema: "",
        tipo: "",
        select: ["Afiche", "Folleto", "Separador", "Flyer"],
        Participante: "",
        fecha: ""
```

```
      },
      show: true
    };
  },
  methods: {
    /* Los métodos creartablaAudios y  elegirBusquedaRapida requieren de Axios para generar búsquedas por método GET
    Una vez se despliegue el servidor en la red de datos con una ip pública se debe copiar y reemplazar por el localhost*/
    creartablaPublicidad() {
      this.axios
        .get(
          "http://localhost:3000/publicidad/" +
            this.form.tema +
            "," +
            this.form.tipo +
            "," +
            this.form.Participante +
            "," +
            this.form.fecha
        )
        .then(res => {
          this.misPublicaciones = res.data;
          this.showSearch = true;
          if( this.misPublicaciones==null){
            this.showModal = true;
            this.misPublicaciones=[];
          }
        })
        .catch(e => {
          this.showModal = true;
          console(e.response);
        });
    },
    /* Limpiamos campos del formulario */
    reset() {
      this.$refs.form.reset();
    },
    /* Método para solicitar una búsqueda por filtros */
    elegirFiltros() {
```

```
      this.VerBotones = false;
      this.VerFiltros = true;
    },
    elegirBusquedaRapida() {
      this.VerBotones = false;
      this.Fast = true;
      this.axios
        .get("http://localhost:3000/publicaciones")
        .then(res => {
          this.misPublicaciones = res.data;
        })
        .catch(e => {
          this.showModal = true;
          console(e.response);
        });
    }
  }
};
</script>
```

Por último en cuanto al manejo de estilos se implementó sass en tres clases: globales, Principal y animate

globales

```
/* ESTILOS GLOBALES
Esta clases permite definir tamaño de letra, colores, tipo de fuente... */

@import 'animate';
body {
    background: #fff;
    font-family: 'Roboto', sans-serif;
    margin: 0;
    padding: 0;
}

a {
    text-decoration: none;
    color: white;
```

```
}

table {
    width: 100%;
    border-collapse: collapse;
    text-align: center;
}

table,
td,
th {
    border: 1px solid rgba(0, 0, 0, 0.2);
    padding: 10px 20px;
    background: white;
}

caption {
    padding: 0.3em;
    color: #fff;
    background: #000;
}

h3 {
    text-align: center;
    font-size: 35px;
    font-weight: normal;
}

.desplegar {
    border: 1px #AA5E17 solid;
}

.SeleccionarTipoBusqueda {
    padding: 10%;
    width: 20px;
    background-color: yellowgreen;
    color: white;
}

btn {
```

```
    cursor: pointer;
    background: #2C8378;
    color: #ffff;
    font-size: 16px;
    text-decoration: none;
    padding: 10px 15px;
    display: inline-block;
}

.Seleccion {
    position: absolute;
    display: flex;
    flex-flow: column;
    justify-content: center;
}

.parrafo,
p {
    font-size: 13px;
}


/* Responsive design */

@media (max-width: 600px) {
    body {
        background: #fff;
        font-family: 'Roboto', sans-serif;
        margin: 0;
        padding: 0;
        font-size: 15px;
    }
    .parrafo,
    p {
        font-size: 10px;
    }
    h3 {
        font-size: 20px;
    }
}
```

IMPLEMENTACIÓN

La implementación del backend y la base de datos se realizó en un servidor linux provisionado por la red de datos UDNET el cual cuenta con las siguientes caracteristicas:

IP Servidor: 10.20.200.180

el acceso al servidor se hace de manera remota por medio de ssh, y es necesario estar conectado a la red de la universidad o a la vpn de esta misma.

Acceso SSH
usuario: ipazud
Contraseña: ohthaeB0ee.2019

A continuación se muestra el usuario y contraseña para el gestor de la base de datos, el cual es MariaDB.

Mysql
Usuario: dbipazud
Contraseña: uoCee3ia.2019

Una vez se tenga acceso al servidor se crea la base de datos con el nombre 'pasantia' y se importa el script sql para la creación de las tablas de la base de datos, para enviar el archivo sql con el script se puede hacer uso del comando SCP, el cual esta brevemente explicado en la siguiente direccion:
https://success.tanaza.com/s/article/How-to-use-SCP-command-on-Windows

este comando también es de utilidad para enviar el archivo de GO al servidor y poder ejecutarlo.

Para la correcta ejecución de la API REST fue necesario instalar Golang en su version 1.13 , agregar las librerías mencionadas en la parte del backend y configurar en GOPATH, para configurar el GOPATH hay que agregar un línea al archivo .bash_profile que se encuentra en la carpeta del usuario del servidor, en el siguiente link está explicado de manera muy sencilla en la seccion de Bash: https://github.com/golang/go/wiki/SettingGOPATH

Una vez realizado todo esto, ejecutamos el archivo main.go en el servidor y este ya estará escuchando las peticiones del frontend, para retornar la información de la base de datos en formato json

Como alternativa de entorno de desarrollo el proyecto está soportado por docker y docker-compose; en el siguiente repositorio de Github podrán encontrar todo el material: https://github.com/Dwan13/BuscadorIPAZUD