Documento para exolicar como se hacen las llamadas a la api

Por ruben Eduardo Gaxiola ochoa

Endpoints

Para el momento de hacer este documento había un total de 11 endpoints en la api que cubren las funcionalidades básicas de el sitio, todos hacen consultas sobre la base de datos de una u otra forma y todos envían y reciben información en forma de json

```
router.get('/', getUsers)
router.post('/login', getUser)
router.post('/signup', signup)
router.post('/crearsala', createsala)
router.get('/entrarsala/:salaid/:iduser', getformulario)
router.post('/enviarresultado', enviarresultados)
router.get('/obtenerresultado/:salaid', recibirresultados)
router.get('/obtenerresultadoalumno/:salaid/:alumnoid', recibirresultadoalumno)
router.get('/obtenerlistaresultados/:iduser', listaparticipaciones)
router.get('/obtenerlistasalas/:iduser', listasalas)
router.get('/obtenerlistaparticipantes/:idsala', usuariosparticipantes)
```

La api tiene una ruta base a la cual se le deben concatenar el resto de la ruta mostrada en los endpoints anteriores

Esa ruta base es http://localhost:3000/api/ , “localhost:3000” se debe cambiar por la ip donde estará corriendo la api un ejemplo de un endpoint completo podría ser

http://localhost:3000/api/obtenerlistaparticipantes/1111, Donde obtenerlistaparticipantes es la acción que se va a realizar y 1111 es el parámetro que se le va a dar, esto quiere decir que se quiere recuperar la lista de participantes de la sala 1111

Los endpoints que no tienen parámetros son endpoints que reciben la información en un objeto json (a excepción del primer endpoint que ni se usa)

Hay algunos endpoints que piden tanto un parámetro como un json

Explicación de cada endpoint

- Login

Esta ya está implementada así que explicare como le hice porque todos los endpoints siguen un método similar

hice un objeto llamado login con dos parámetros, username y password, estos nombres son los mismos que tienen la base de datos

```
const[Login] = useState({
    username: "",
    password: ""
})
```

```
Login.username = username;
Login.password = password;
const datosUser = await axios.post(apiUrl + '/api/login', Login)
console.log(datosUser.statusText);
//console.log(datosUser);
if(datosUser.data.length > 0){

  guardarDatos(datosUser.data[0]);
}else{
  alert("Nombre de usuario o contraseña incorrectos")
}
```

Le asigne valores a cada parámetro y llame a la api con el método axios.post, el parámetro apiUrl es una variable global que está en la pagina y esta se importa de esta forma `const apiUrl = import.meta.env.VITE_API_URL;`

Al parámetro apiurl le concatenas /login/acción y en un parámetro separado le doy el objeto Login, que es donde esta la información que la api desea recibir, si encuentra al usuario te devolverá un objeto json con toda la información del usuario que esta iniciando.

La información se extrae de esta forma, los parametros tienen los mismos nombres que en el json

```
const guardarDatos = (userData) => {
  console.log(userData)
  localStorage.setItem('isLoggedIn', 'true');
  localStorage.setItem('iduser', userData.id);
  localStorage.setItem('username', userData.username);
  localStorage.setItem('correo', userData.correo);
  localStorage.setItem('password', userData.password);
  localStorage.setItem('tipousuario', userData.tipousuario);

  if(userData.tipousuario == 1){
    alert("Alumno iniciado")
    navigate("/codigo")
  }else{
    alert("Maestro Iniciado")
    navigate("/crearcuestionario")
  }
}
```

- Registrar

Esta también esta implementada y sigue un proceso similar al anterior

```
const[Register] = useState({
  username: "",
  correo: "",
  password: "",
  tipousuario: localStorage.getItem("rol"),
})

const adduser = () =>{
  axios.post(apiUrl + '/api/signup/', Register).then(() => {
    alert(`Usuario ${Register.username} ha sido registrado, por favor inicie sesion con su usuario`)
    navigate("/login")
  })
 console.log(Register)
}
```

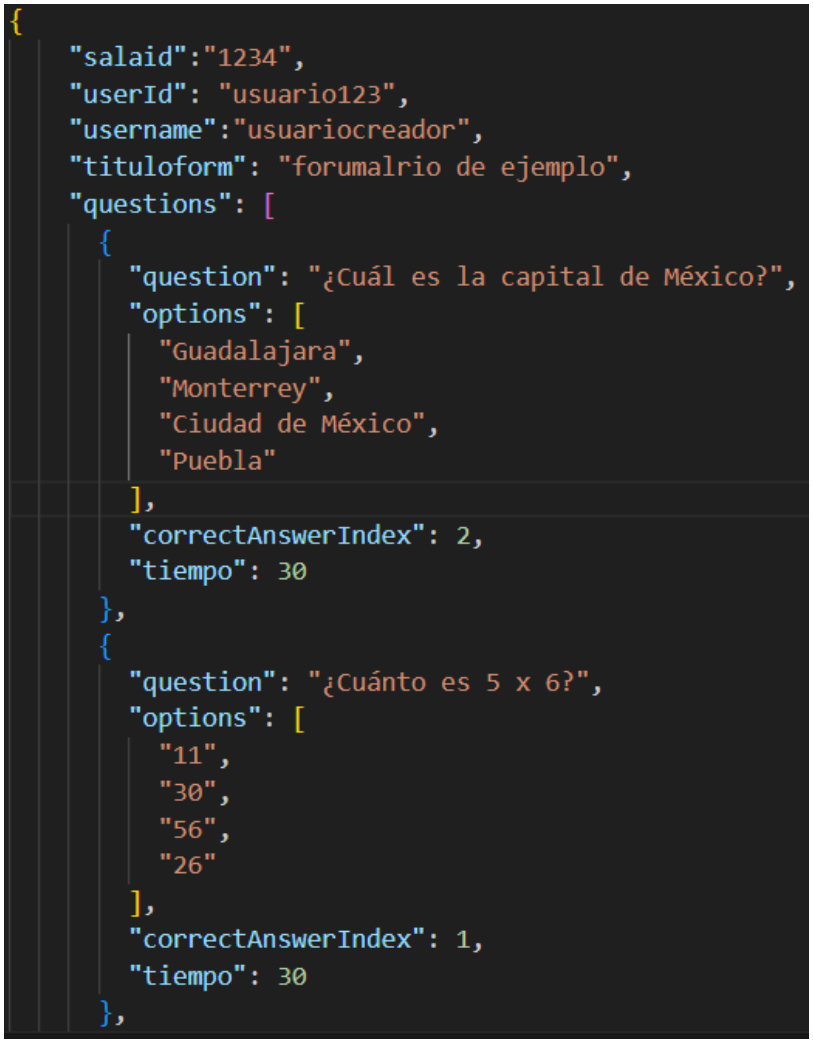
Es el mismo proceso

A partir de la siguiente ya no están implementadas y tratare de explicar lo mejor posible como funcionan y como se pueden usar

- Crear sala

```
router.post('/crearsala', createsala)
```

Este endpoint espera un objeto de tipo json con toda la información perteneciente a la sala, concretamente el objeto que se espera es así



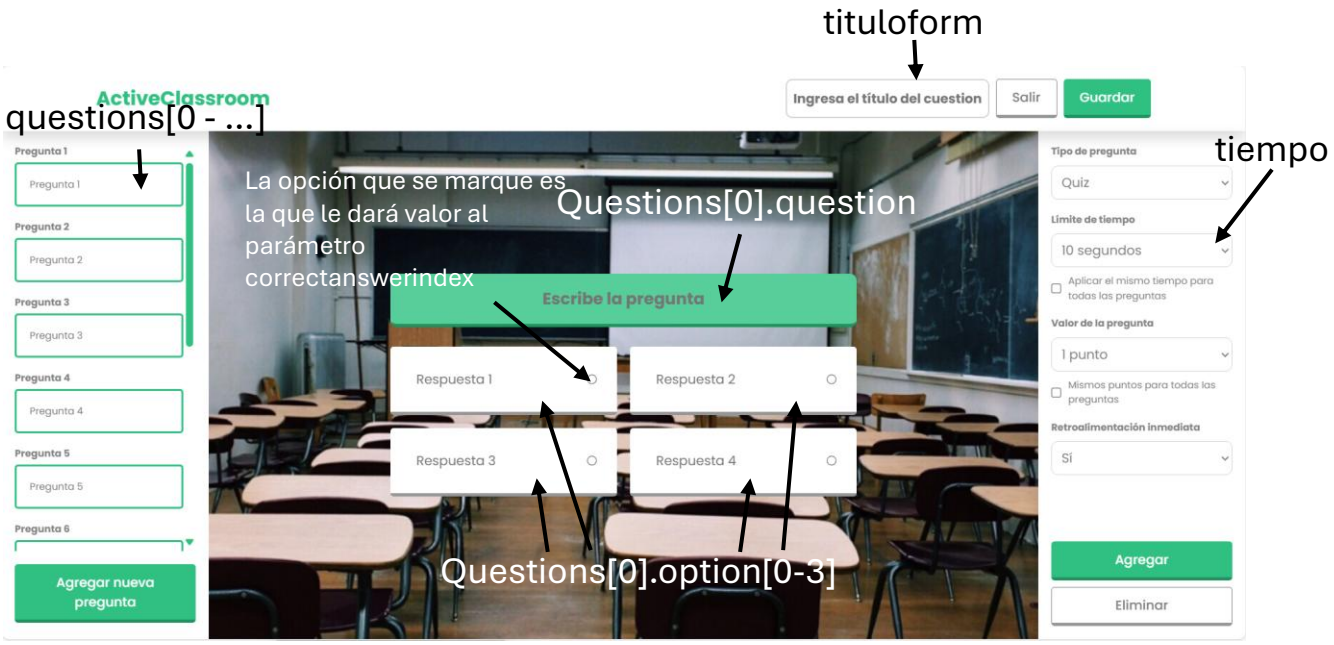
```
{
    "salaid":"1234",
    "userId": "usuario123",
    "username":"usuariocreador",
    "tituloform": "forumalrio de ejemplo",
    "questions": [
      {
        "question": "¿Cuál es la capital de México?",
        "options": [
          "Guadalajara",
          "Monterrey",
          "Ciudad de México",
          "Puebla"
        ],
        "correctAnswerIndex": 2,
        "tiempo": 30
      },
      {
        "question": "¿Cuánto es 5 x 6?",
        "options": [
          "11",
          "30",
          "56",
          "26"
        ],
        "correctAnswerIndex": 1,
        "tiempo": 30
      },
```

Salaid: es el código de la sala el cual también hace de llave primaria en la tabla
userId: es la id del maestro
username: es el nombre del creador
Estos dos parámetros se obtienen de los datos de sesión del login
tituloform: es el titulo del formulario
questions: es un arreglo de objetos en el que cada objeto tiene sus propios parámetros, es aquí donde se guarda la información de cada pregunta.
questions.quiestion: es la pregunta en si
questions.options: son las posibles respuestas de la ptegunta.
questions.correctAnswerIndex: es el índice de la pregunta correcta, estos índices van del 0 al 3,

questions.tiempo: es el tiempo asignado para esa pregunta.

Nota: los nombres que ves ahí deben ser los mismos que asignes en los parámetro en el código, si los nombres son diferentes no va a funcionar así que apegate a esos nombres.

- De donde se saca la información para armar el json anterior, voy a tratar de explicarlo para que quede claro

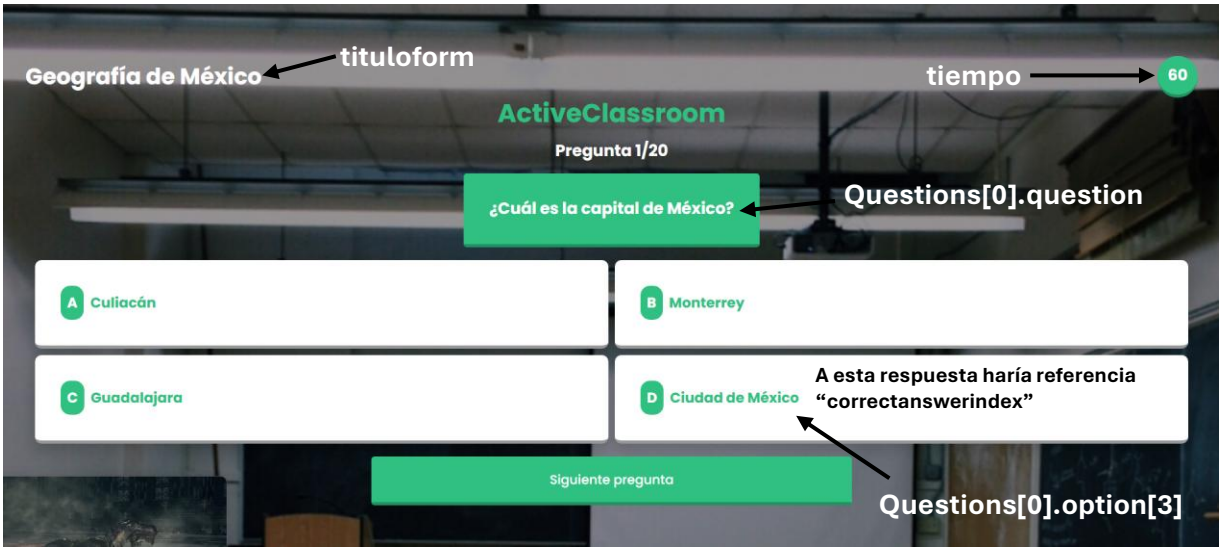


El resto de parámetros se sacan de datos de sesión o de variables que se guardan el en local storage del navegador como la id de la sala,
El tipo de pregunta no necesariamente cambia la estructura del json, ya que solo reducira la cantidad de opciones

- Entrar a sala

```
router.get('/entrarsala/:salaid/:iduser', getformulario)
```

Este endpoint no pide un objeto de tipo json, pide dos parámetros en la url, la id de la sala y la id del usuario que esta solicitando la sala, si la llamada a la api se realiza correctamente recibirás un objeto json similar al del endpoint anterior, son los mismos parámetros así que para no repetirme voy a mostrar como se podría utilizar la información de este json para mostrar en la pagina.



Así seria mas o menos, salaid y usuarioid se usan para otras cosas como hacer consultas a la base de datos y así pero como tal nunca se muestran en el formulario, no hace falta.

- Enviar resultados

```
router.post('/enviarresultado', enviarresultados)
```

Este endpoint si requiere un objeto de tipo json y se llama una vez el alumno termina el cuestionario, se recopilaran todas las respuestas del alumno. El objeto que se espera es este


```
{
  "salaid": "1111",
  "userId": "2312",
  "username": "tata",
  "participanteid": "001",
  "calificacion": " 7",
  "tituloform": "forumalrio de ejemplo",
  "questions": [
    {
      "question": "¿Cuál es la capital de México?",
      "options": [
        "Guadalajara",
        "Monterrey",
        "Ciudad de México",
        "Puebla"
      ],
      "correctAnswerIndex": 2,
      "electionindex": 2,
      "tiempo": 33
    },
    {
      "question": "¿Cuánto es 5 x 6?",
      "options": [
        "11",
        "30",
        "56",
        "26"
      ],
      "correctAnswerIndex": 1,
      "electionindex": 1,
      "tiempo": 33
    },
```


Como ves, es similar al json del formulario creado en el endpoint crear sala, pero tiene algunos parámetros extra, solo explicare estos parámetros nuevos
participanteid: el id del alumno que envia los resultados
calificacion: la calificación que recibió el alumno
De ahí pasamos al objeto questions, en cada objeto se añaden dos parámetros extra
electionindex: es la opción que selecciono el alumno
tiempo: es el tiempo que tardo el alumno para responder la pregunta

De nuevo, los nombres deben ser exactamente iguales en el json que te mostré y el objeto que crees en el código, respetando tanto tipo de dato (si es un entero o un string) así como respetar mayúsculas y minusculas

- Obtener resultado

```
router.get('/obtenerresultado/:salaid', recibirresultados)
```

Este endpoint lo ejecuta el profesor, una vez se asegure que todos los alumnos hayan terminado el formulario, como ves solo pide un parámetro, la id de la sala que creo el profesor, al hacer la petición recibirás un json el cual se divide en dos partes


```
[
    [
        {
            "ID de sala": 1111,
            "ID de formulario": 11,
            "ID participacion": 1,
            "Nombre participante": "Ruben",
            "Calificacion": 7,
            "Ruta del resultado": "C:\\Users\\ruben\\Documents\\GitHub\\active-classroom\\back\\forms\\resultados\\resultado_formulario_001_1111.json",
            "Respuesta pregunta 0": 2,
            "Tiempo pregunta 0": 33,
            "Respuesta pregunta 1": 1,
            "Tiempo pregunta 1": 33,
            "Respuesta pregunta 2": 2,
            "Tiempo pregunta 2": 33
        },
        {
            "ID de sala": 1111,
            "ID de formulario": 11,
            "ID participacion": 2,
            "Nombre participante": "Manuel",
            "Calificacion": 10,
            "Ruta del resultado": "C:\\Users\\ruben\\Documents\\GitHub\\active-classroom\\back\\forms\\resultados\\resultado_formulario_002_1111.json",
            "Respuesta pregunta 0": 2,
            "Tiempo pregunta 0": 33,
            "Respuesta pregunta 1": 1,
            "Tiempo pregunta 1": 33,
            "Respuesta pregunta 2": 3,
            "Tiempo pregunta 2": 33
        },
        {
            "ID de sala": 1111,
            "ID de formulario": 11,
            "ID participacion": 3,
            "Nombre participante": "Juan",
            "Calificacion": 0,
            "Ruta del resultado": "C:\\Users\\ruben\\Documents\\GitHub\\active-classroom\\back\\forms\\resultados\\resultado_formulario_003_1111.json",
            "Respuesta pregunta 0": 1,
            "Tiempo pregunta 0": 33,
            "Respuesta pregunta 1": 3,
            "Tiempo pregunta 1": 33,
            "Respuesta pregunta 2": 2,
            "Tiempo pregunta 2": 33
        }
    ],
```


La primera parte es un arreglo que contiene información de todos los alumnos que participaron en el formulario, se ven datos como la id de la sala, id del formulario el nombre del alumno la ruta (esta es información sacada de la base de dato, todas salvo el nombre del alumno puedes ignorarlo) y los mas importantes, se muestran la elección del alumno a cada pregunta y el tiempo que tardo para responderla, es de aquí de donde sacas los datos de los resultados de cada alumno, además esto se puede complementar con la segunda parte de este json

```
    {
        "salaid": "1111",
        "userId": "2312",
        "tituloform": "forumalrio de ejemplo",
        "questions": [
            {
                "question": "¿Cuál es la capital de México?",
                "options": [
                    "Guadalajara",
                    "Monterrey",
                    "Ciudad de México",
                    "Puebla"
                ],
                "correctAnswerIndex": 2
            },
            {
                "question": "¿Cuánto es 5 x 6?",
                "options": [
                    "11",
                    "30",
                    "56",
                    "26"
                ],
                "correctAnswerIndex": 1
            },
            {
                "question": "¿Cuál es el océano más grande?",
                "options": [
                    "Atlántico",
                    "Índico",
                    "Ártico",
                    "Pacífico"
                ],
                "correctAnswerIndex": 3
            }
        ]
    }
]
```


Como ves no es mas que una copia del formulario, exactamente el mismo que envias en el endpoint crear sala y el que recibe el alumno al entrar al formulario usando esta parte y las respuestas de los alumnos en la primera sección del json puedes comparar la respuesta del alumno y el índice de la respuesta correcta para saber si el alumno respondió correctamente

- Obtener resultados de alumno

```
router.get('/obtenerresultadoalumno/:salaid/:alumnoid', recibirresultadoalumno)
```

Este endpoint es similar al anterior pero es para el alumno, una vez termine el formulario, el alumno podrá ver sus resultados con la información sacada de este endpoint, pide la id de la sala y la id del alumno y nos devuelve este objeto


```
{
    "id": 1,
    "iduser": 1,
    "idsala": 1111,
    "calificacion": 7,
    "rutaresultados": "C:\\Users\\ruben\\Documents\\GitHub\\active-classroom\\back\\forms\\resultados\\resultado_formulario_001_1111.json"
}
```


Le falta información en la semana lo arreglo xDDDD no me acordaba que lo dejé incompleto

- Creo que aquí ya explique los mas importantes quedan otros tres endpoints pero no tienen nada especial, ya que son gets, ósea solo envia un parámetro y recibes un json con la información, si alcanzamos a implementar eso pues ya lo explico pero primero hay que implementar los que explique que creo yo son los mas importantes

Animo que a partir de aquí solo tenemos 3 semanas para hacer que esto funcione