Presentacion Tecnica - JocQuiz

Hecho por : Josthyn Loma y Kevin Hoyos

Problemas y soluciones

Problemas:

Hemos tenido varios problemas durante el proceso , los más destacados eran , o que desde vue no enviaba bien la información a la API o en symfony había problemas a la creación , normalmente salen errores al hacer fetch con el método POST. Salia este error porque no construía bien el json que tendría que recibir la API o desde backend pedía una json bastante complejo

Soluciones:

Encontraba una forma coherente en la que pueda enviar bien el json y utilizando la función JSON.stringify(array) para que se pueda enviar o en symfony(backend) para solucionarlo buscaba otra manera en la que pueda recibir el json que envía fetch correctamente

Problemas Actuales:

No está bien hecho el diseño, el administrador no funciona correctamente, y no hay control para introducir datos en crearQuiz y no comprueba si un campo de texto está vacío

Aspectos Técnicos - Frontend

- **Registro** : Está compuesta por dos archivos , una que está toda la información que saldrá por pantalla y que enviará los datos que introducimos con un POST y otro archivo en la que hará que que se muestre por pantalla
- **Login:** Con un fetch hará un POST que dará un mensaje si esta logueado o no, si esta logueado con pinia guardamos toda la información del usuario para crear una sesión
- **Admin**: Con un fetch POST correctamente dará un mensaje si eres admin o no, comprueba con un if si el mensaje llegado desde symfony es correcto o no, y guardará la información a pinia
- **Perfil**: Recoge toda la información de pinia para que se refleje en los campos de texto y con un método POST desde VUE y método PUT desde symfony se hará el cambio si el usuario desea cambiar su información.
- **Quiz Creados:** Cojera la información de los quiz que ha creado el usuario con un fetch GET y con un método DELETE podrá eliminar el quiz con la API que ha creado desde backend

Frontend

- **Quiz Jugados:** Fetch GET para recoger la información de los quiz que ha realizado el usuario
- **Jugar Quiz**: Fetch GET que recogerá las preguntas dependiendo del quiz y al terminar hara un POST a la api para enviar las respuestas
- **Crea Quiz** : Con un fetch POST conseguimos enviar todas las preguntas, respuesta y su correspondiente resultado. Para hacer bien este POST se ha tenido que construir la estructura del json y en el body escribir cómo se enviará sin necesidad de la variable datosEnvio
- **Página Inicia**: Compuesta por varios Gets que hara mostrar mejores jugadores,mejores cuestionarios y que tipos de cuestionarios hay
- **Mostrar Quiz:** Hará un get para recoger todos los cuestionarios de cierto tema

```
export default {
    data() {
      return {
        usuario: "",
        titulo: "",
        respues:
            [
            {
            enunciado:'',
            respuestas:[{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''}]
            },
            {
            enunciado:'',
            respuestas:[{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''}]
            },
            {
            enunciado:'',
            respuestas:[{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''}]
            },
                {
            enunciado:'',
            respuestas:[{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''}]
            },
            {
            enunciado:'',
            respuestas:[{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''},{respuesta:'',estado:''}]
            }
```

```
 computed: {
    ...mapStores(sessioStore)
}, beforeMount() {
    //MIRO EL LOCAL STORAGE, SI HAY ALGO LO CARGO
    this.Usuario = this.sessioStore.get.username;
    this.estado = this.sessioStore.get.estadoLogin;
    this.apellido = this.sessioStore.get.apellido;
    this.Correo = this.sessioStore.get.correo;
    this.Contrasena = this.sessioStore.get.contrasena
    this.usuario = this.sessioStore.get.idUser;
},methods: {

    crearQuiz() {

        //const respues = JSON.stringify(this.respues);

        fetch(`http://proyectefinaljoskevback.alumnes.inspedralbes.cat/anadir/quiz` , {
            method: 'POST',
            body: JSON.stringify({usuario:this.usuario,titulo: this.titulo,preguntas: this.respues}),
          }).then(res => {
            return res.json();
          }).then(data => {
          });
        }
}
```

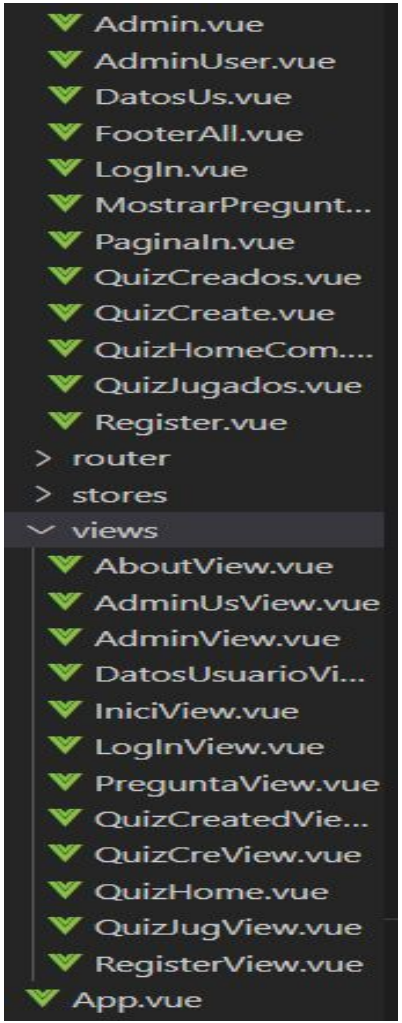
```
<div id="f" v-if="this.estado === true" class="contenidor">
    <h4 style="color:yellow">Titulo de Quiz: </h4>
    <input type="text" name="titulo" v-model="titulo"  class="info form-control" id="titulo" >

        <div class="pregunta" v-for="(item,indexPR) in 5" :key="indexPR">
        Introduce 0 si es falso i 1 es verdadero
            <h5>Pregunta:</h5>
            <input type="text" v-model="this.respues[indexPR].enunciado" :name="'p' + indexPR"  class="info form-control" id="preguntas" >
             <div class="respuestas" v-for="(res,indexRESP) in 5" :key="indexRESP">
                <h5>Respuesta:</h5>
                <input type="text" v-model="this.respues[indexPR].respuestas[indexRESP].respuesta"  :name="'re' + indexI + '_' + indexJ "   class="info form-control" id="resp
                <h5>Estado:</h5>
                <input id="estado" type="text" v-model="this.respues[indexPR].respuestas[indexRESP].estado"  :name="'e' + indexI + '_' + indexJ "   class="info form-control">
            </div>
        </div>
        <button class="btn btn-warning" @click="crearQuiz()">crear quiz</button>
</div>


emplate>
```

```
},methods: {
    eliminarQuiz(id) {
        fetch(`http://proyectefinaljoskevback.alumnes.inspedralbes.cat/quiz/eliminar/${id}` , {
            method: 'DELETE',
          }).then(res => {
            return res.json();
          }).then(data => {
          });
    }
},
```

```
 mounted () {
    fetch(`http://proyectefinaljoskevback.alumnes.inspedralbes.cat/preguntas/${this.$route.params.id}`)
    .then(res => res.json())
    .then((data) => {
      this.myjson = data;
      this.Id = this.sessioStore.get.idUser;
    });
  },
}
```




```vue
<script >
import { RouterLink, RouterView } from 'vue-router'
import MostrarPregunta from '../components/MostrarPregunta.vue'
import Heather from '../components/Heathers/HeatherAll.vue'
export default {
  components: {
    MostrarPregunta,
    Heather
  }
}
</script>

<template>

  <main>
    <heather/>
    <mostrar-pregunta/>
  </main>
</template>
```

Aspectos Técnicos - Backend

QuizController->Función de añadir quiz: recojo los datos del quiz y los voy desglosando para poder añadirlos en las tres diferentes tablas que intervienen (quiz, preguntas y respuestas).

QuizController->Función quizPuntuacion: recojo de la BBDD los quizs jugados de un usuario y con esta información obtengo la mejor puntuación que conseguido en los mismos a través de un SELECT con varios JOINs.

QuizController-> Función quizMasJugados: recojo con un SELECT un vez el id de los quiz que han jugado un partida, calcúlo con un SELECT el número de partidas y halló las 5 primeras.

PartidaController-> Función añadir partida: recojo los datos y calculo la puntuación, añado la partida, encuentro la id de la partida, la id de las preguntas con las respuestas y añado respuestasPartida.