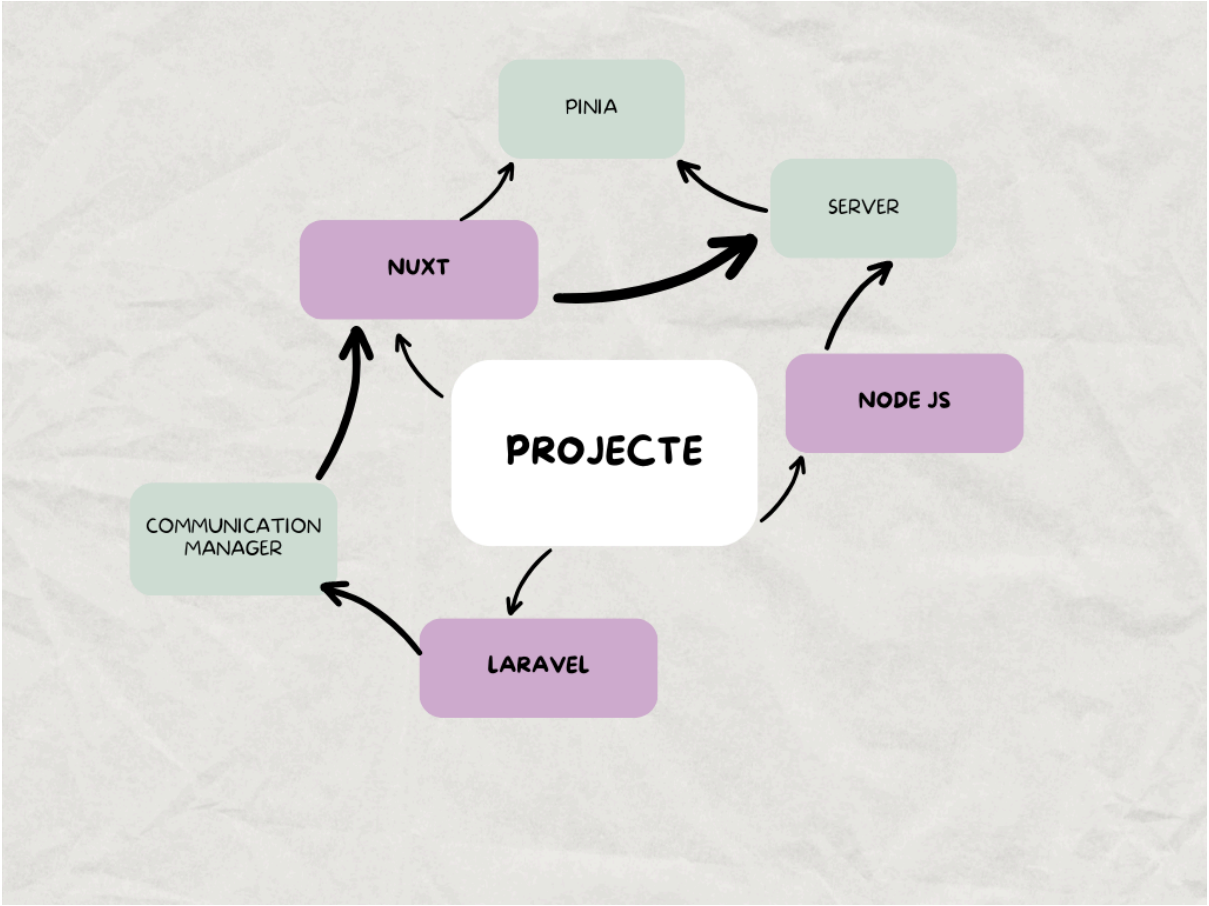
1. **Arquitectura de l'aplicació**:
   En la nostra aplicació fem us del nuxt per la part del front amb el pinia, també utilitzem el node per fer servir els Sockets, y tambe el laravel per a la fiabilitat de les preguntes y respostes en el joc. Al nuxt fem servir rooms per el jugadors y anem enviant cambis al pinia desde el server.



2. **Rutes de l'aplicació:**

- Descripció:
  - Que fa: Enviar atac el que fa es enviar les dades al back per que hem torni les dades de les preguntes. Utilitzo el metode de communication manager per la organitzacio de dades.
  - Método HTTP: [POST]
  - URL de la ruta: [http://localhost:8000/api/enviar-atac
- Parámetros que recibe:
  - [name] ([String]): [es el nombre del pais seleccionado]
  - [IdUser]([Integer]): [es el id del usuario que envia el ataque]
- Parámetros u objetos que retorna:
  - Respuesta en caso de éxito (response.data]):
    - [data] ([JSON]): [Devuelve los datos con la pregunta y las respuestas]
  - Respuesta en caso de error ([error]):
    - [error] ([String]): [devuelve el error]
- Servicio del backend:

- [Nombre del servicio: Laravel]

```
async enviarAtac(name, paisId, idUser) {

      this.cambiarAccion("Atacando");
      try {
        const data = await enviarAtac(name, idUser);

        this.pregunta = {
          id: data.pregunta.id,
          pregunta: data.pregunta.pregunta,
          respuesta_a: data.pregunta.respuesta_a,
          respuesta_b: data.pregunta.respuesta_b,
          respuesta_c: data.pregunta.respuesta_c,
          respuesta_d: data.pregunta.respuesta_d,
        };

        this.app.setPaisSeleccionado(paisId);
```

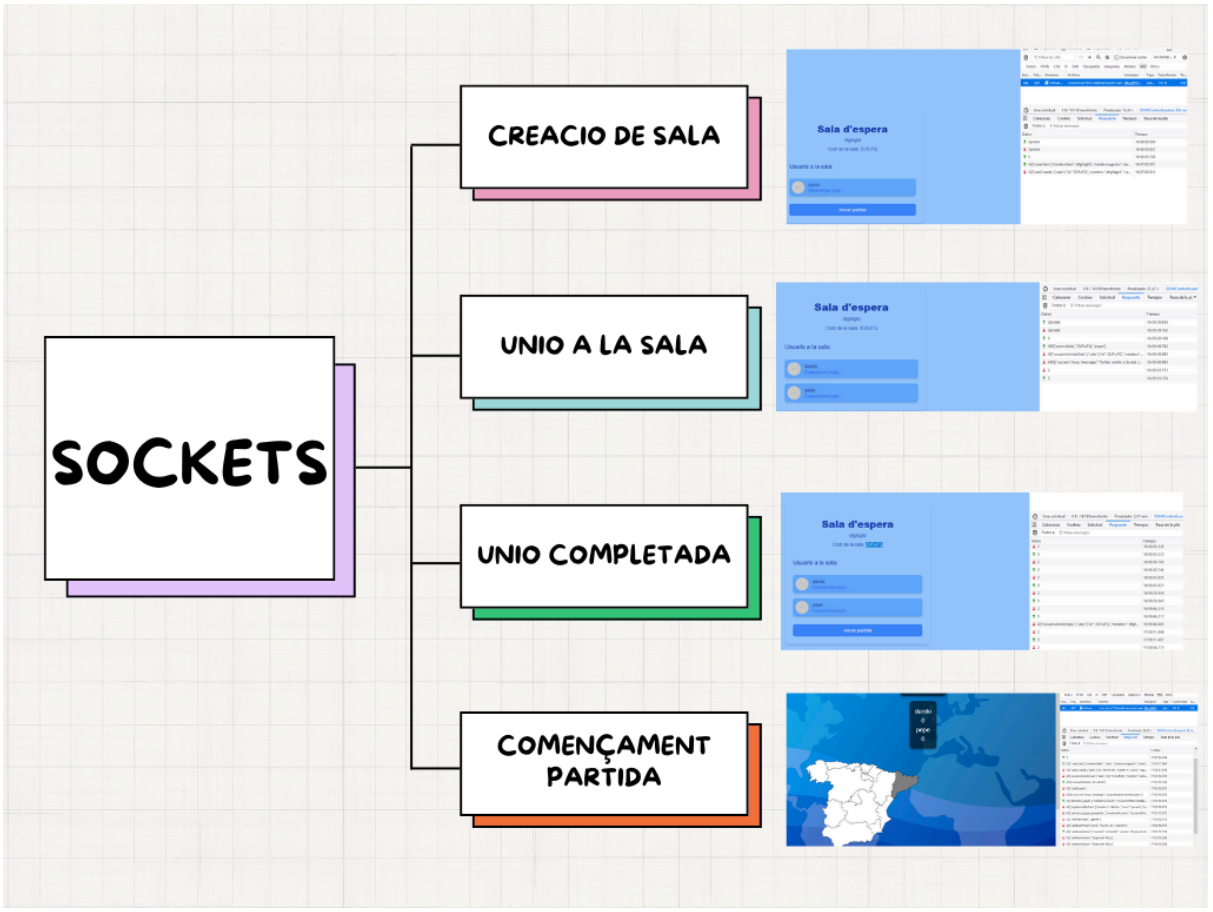
```
export async function enviarAtac(name, idUser) {
    try {
        const response = await fetch(`${url}/api/enviar-atac`, {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify({
                name: name,
                idUser: idUser,
            }),
        });

        if (!response.ok) {
            throw new Error(`Error en la solicitud:
${response.status}`);
        }

        const data = await response.json();
        console.log("Respuesta del servidor:", data);
        return data; // Retornar los datos de respuesta
```

```
    } catch (error) {
        console.error("Error en la solicitud:", error);
        throw error; // Propagar el error hacia arriba
```

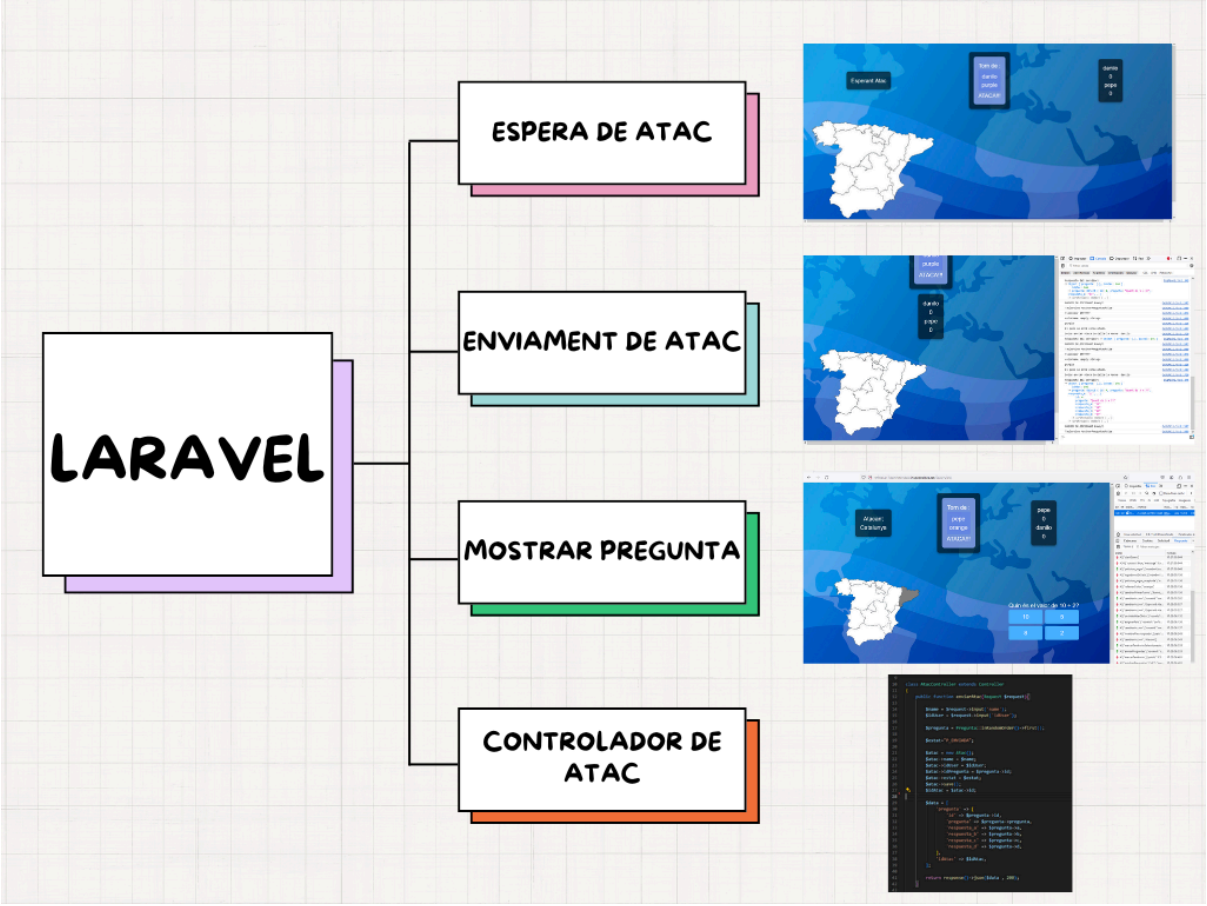
- Descripció:
  - Que fa: El que fa el emit marcarTerritorioSeleccionado es enviar al servidor el id de la sala y el paisID que he seleccionat, per pintarlo de color gris.
  - Método HTTP: [POST]
  - URL de la ruta: `socket.emit('marcarTerritorioSeleccionado'`
- Parámetros que recibe:
  - [roomId] ([Integer]): [es el id de la sala en el que se esta jugando la partida]
  - [paisId] ([integer]): [es el id de el pais que acabo de seleccionar para poder identificarlo y poder cambiar de color.]
- Parámetros u objetos que retorna:
  - no devuelve datos, efectua cambios en el pinia de mi cliente.
- Servicio del backend:
  - [Nombre del servicio: node]

```
socket.emit('marcarTerritorioSeleccionado', { roomId: this.app.sala.id,
paisId: paisId });



 socket.on('marcarTerritorioSeleccionado', ({ roomId, paisId }) => {
    console.log("Territorio SELECCIONADO SERVER");
    io.to(roomId).emit('marcarTerritorio', { paisId });
  });


socket.on('marcarTerritorio', ({ paisId }) => {
  const appStore = useAppStore();
  console.log(`¡Territorio ${paisId}! con color gris`);
  appStore.setPaisSeleccionado(paisId);
  const paisElement = document.getElementById(paisId);
  if (paisElement) {
    paisElement.style.fill = 'grey';
  }
});
```

3. **Esquema d’esdeveniments de comunicació (sockets):**



4. **Esquema d’esdeveniments de comunicació (sockets):**



5. Esquema de components de frontend

- pages/: Carpeta clave para las rutas y vistas principales de tu aplicación.
- public/: Carpeta para archivos estáticos que serán servidos directamente.
- server/: Parte del backend, no directamente relacionado con el frontend.
- services/: Podría contener lógica compartida entre frontend y backend, como llamadas a APIs.
- stores/: Carpeta utilizada para Vuex stores (gestión del estado).
- utils/: Carpeta para utilidades y funciones helper.
- app.vue: Archivo raíz del componente de la aplicación.

6. Documentació de codi frontend:

```
<template>
  <!-- Contenedor principal con imagen de fondo -->
  <div class="w-screen h-screen flex items-center justify-center
bg-center bg-cover bg-no-repeat" style="background-image:
url('/mapaRisk.png');">
    <div class="login-container" v-if="!nombreEscrito">
      <h2>Risk Math</h2>
      <div class="input-container">
        <input v-model="nombreUsuario" />
        <label for="nombreUsuario">Nombre de usuario</label>
      </div>
      <!-- Botones para crear o unirse a una sala -->
      <button @click="crearSala">Crear sala</button>
      <button @click="unirseSala">Unirse a sala</button>
      <!-- Enlace para administrar preguntas -->
      <nuxt-link to="/AdminPreguntes" class="adminPreg">Administrar
Preguntes</nuxt-link>
      <!-- Icono de información con funcionalidad de popup -->
      <img src="../public/info_icon.png" @click="popupInfo"
class="info-icon">
      <div class="superpuesto" id="superpuesto">
        <button @click="popoffInfo" class="poppup_btn">x</button>
        <p>En "MultipliCAT", dos jugadors competeixen responent
preguntes sobre mesures per conquerir
          territoris. Cada
          territori té una pregunta sobre sistemes de mesures.
L'objectiu: guanyar responent correctament i
          dominar el món
          demostrant coneixements en unitats i conversions.</p>
      </div>
    </div>
  </div>
</template>

<script>
import { socket } from '@/utils/socket.js'; // Importa la configuración
del socket
import { useAppStore } from '../stores/app'; // Importa el store de la
aplicación

export default {
```

```
  data() {
    return {
      nombreUsuario: null, // Nombre de usuario introducido
      nombreEscrito: false, // Flag para saber si el nombre ha sido
escrito
      app: useAppStore(), // Instancia del store
      user: "", // Usuario actual
      ruta: 'http://localhost:8000', // Ruta base para las peticiones
      countdown: 0, // Contador
      mostrarContador: false, // Flag para mostrar el contador
      link: '', // Enlace para la sala
    };
  },
  methods: {
    // Muestra el popup de información
    popupInfo() {
      var superpuesto = document.getElementById("superpuesto");
      superpuesto.classList.add("mostrar");
    },
    // Oculta el popup de información
    popoffInfo() {
      var superpuesto = document.getElementById("superpuesto");
      superpuesto.classList.remove("mostrar");
    },
    // Crea una nueva sala y navega a la página de creación de sala
    crearSala() {
      if (this.nombreUsuario == null) {
        alert("Introduce un nombre de usuario");
      } else {
        let store = useAppStore();
        store.setNombre(this.nombreUsuario);
        this.$router.push({ name: 'CrearSala' });
      }
    },
    // Se une a una sala existente y navega a la página de unión de
sala
    unirseSala() {
      if (this.nombreUsuario == null) {
        alert("Introduce un nombre de usuario");
      } else {
        let store = useAppStore();
        store.setNombre(this.nombreUsuario);
        this.$router.push({ name: 'UnirseSala' });
```

7. Documentació de codi backend:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Session;
use Illuminate\Http\Request;
use App\Models\Atac;
use App\Models\Pregunta;
use Illuminate\Support\Facades\Log;

class AtacController extends Controller
{
    /**
     * Método para enviar un ataque.
     * Este método crea un nuevo ataque con una pregunta seleccionada aleatoriamente.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\JsonResponse
     */
    public function enviarAtac(Request $request){
        // Obtiene el nombre y el ID de usuario de la solicitud
        $name = $request->input('name');
        $idUser = $request->input('idUser');

        // Selecciona una pregunta aleatoria
        $pregunta = Pregunta::inRandomOrder()->first();

        // Estado inicial del ataque
        $estat = "P_ENVIADA";

        // Crea un nuevo ataque y lo guarda en la base de datos
        $atac = new Atac();
        $atac->name = $name;
        $atac->idUser = $idUser;
        $atac->idPregunta = $pregunta->id;
        $atac->estat = $estat;
        $atac->save();

        // Obtiene el ID del ataque recién creado
```

```
        $idAtac = $atac->id;

        // Estructura de datos para la respuesta JSON
        $data = [
            'pregunta' => [
                'id' => $pregunta->id,
                'pregunta' => $pregunta->pregunta,
                'respuesta_a' => $pregunta->a,
                'respuesta_b' => $pregunta->b,
                'respuesta_c' => $pregunta->c,
                'respuesta_d' => $pregunta->d,
            ],
            'idAtac' => $idAtac,
        ];

        // Devuelve la respuesta JSON con el estado 200
        return response()->json($data, 200);
    }

    /**
     * Método para cambiar el estado de un ataque.
     * Este método actualiza el estado de un ataque basado en el
resultado de la pregunta.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\JsonResponse
     */
    public function cambiarEstadoAtaque(Request $request){
        // Obtiene el resultado y el ID del ataque de la solicitud
        $resultado = $request->input('resultado');
        $ataqueID = $request->input('ataqueId');

        // Busca el ataque por ID
        $ataque = Atac::find($ataqueID);

        // Verifica si el ataque existe
        if ($ataque) {
            // Actualiza el estado del ataque basado en el resultado
            if ($resultado) {
                $ataque->estat = "ACERTADA";
            } else {
                $ataque->estat = "FALLADA";
            }
```

```
            // Guarda los cambios en la base de datos
            $ataque->save();

            // Estructura de datos para la respuesta JSON
            $data = [
                'mensaje' => 'Estado del ataque cambiado exitosamente',
                'id' => $ataque->id,
            ];

            // Devuelve la respuesta JSON con el estado 200
            return response()->json($data);
        } else {
            // Estructura de datos para la respuesta de error
            $data = [
                'mensaje' => 'Error: El ataque no existe o no pertenece
al usuario actual',
                'id' => $ataque,
            ];

            // Devuelve la respuesta JSON con el estado 400
            return response()->json($data, 400);
        }
    }
}
```

Llista de Confrontament

#	Coneixement	Molt	Bastant	Poc	Gens
1	Sé documentar l'arquitectura del sistema	x			
2	Sé documentar les rutes de l'aplicació	x			
3	Sé documentar els esdeveniments dels sockets	x			
4	Sé documentar l'esquema de la base de dades	x			
5	Sé crear un esquema de components	x			
6	Sé formatar i comentar codi en frontend	x			
7	Sé formatar i comentar codi en backend	x			
8	Entenc les interioritats del desplegament	x			
9	Sé localitzar i modificar el CSS de qualsevol element	x			