ADI

Trabajo grupal

2023 / 24

Mario Davó
Carlos Pineda

Índice

Descripción del proyecto

Este proyecto se ha desarrollado utilizando dos enfoques diferentes: Vue.js, un framework de JavaScript ampliamente utilizado, y Vanilla JS, que representa la implementación de JavaScript puro sin el uso de frameworks. A lo largo de esta documentación, analizaremos las diferencias clave entre la versión Vue.js y la implementación en Vanilla JS. Esta comparación nos permitirá comprender de manera más profunda las ventajas y desventajas de cada enfoque, destacando las peculiaridades y la eficiencia de Vue.js en relación con la simplicidad y flexibilidad de Vanilla JS.

Api de OpenTrivia

La API de Open Trivia Database (OpenTDB) permite realizar solicitudes sin necesidad de un token de API, utilizando en su lugar un sistema de control basado en la dirección IP para limitar la frecuencia y cantidad de solicitudes. Mientras que el token de API no es obligatorio, su utilidad radica en evitar la repetición de preguntas en las respuestas proporcionadas por la API. Al incluir el token en las solicitudes, se garantiza que las preguntas entregadas no se repitan dentro de un mismo conjunto de solicitudes, brindando así una experiencia de juego más variada y dinámica. En resumen, aunque el token no es esencial para acceder a la API, su uso mejora la calidad de las respuestas al evitar duplicaciones, proporcionando así una experiencia más enriquecedora para los usuarios de la aplicación trivia.

Interacción con la API

Para interactuar con la Open Trivia DB API, el primer paso es utilizar la URL proporcionada para realizar las llamadas correspondientes. Si se desea evitar la repetición de preguntas, se puede emplear un Session Token, una clave única que ayuda a rastrear las preguntas ya recuperadas. Al anexar un Session Token a una llamada API, se garantiza que no se obtendrán preguntas duplicadas durante la sesión. En caso de agotar todas las preguntas posibles en la base de datos para un Token específico, la API devuelve un “Response Code”, indicando que es necesario “Resetear” el Token para borrar el historial pasado o solicitar uno nuevo.

Los “Response Codes” acompañan cada llamada API y ofrecen información sobre el estado de la operación. Por ejemplo, el Code 0 señala un retorno exitoso, el Code 1 indica que no hay suficientes preguntas para la consulta, el 3 indica que el token no es válido y el 5 que hemos superado el límite de peticiones. Estos son los únicos códigos de respuesta que hemos usado en nuestro proeycto. Además, la API implementa límites, como la restricción de una categoría por llamada y un máximo de 50 preguntas por solicitud.

Además, para manejar posibles caracteres especiales o Unicode en las preguntas, la API devuelve resultados en un formato codificado. Se puede especificar el formato de codificación deseado en la URL. Asimismo, se proporcionan herramientas adicionales, como la búsqueda de categorías, el recuento de preguntas en una categoría específica y el recuento global de preguntas en la base de datos.

Es crucial tener en cuenta las limitaciones, como la expiración de Session Tokens después de 6 horas de inactividad y el límite de una solicitud cada 5 segundos por dirección IP para evitar excesivas peticiones. En conjunto, estas consideraciones y funcionalidades permiten una interacción efectiva y controlada con la Open Trivia DB API.

Proyecto en Vue

El proyecto desarrollado en Vue.js presenta una estructura simple con dos vistas principales. En la vista inicial, los usuarios son recibidos con un desplegable que les permite elegir la categoría de preguntas a las que desean enfrentarse, seguido de un botón que inicia la partida. Una vez en la vista

de la partida, una carta visualiza la pregunta, cuyo color se adapta de manera dinámica según la categoría seleccionada. Este enfoque visual distintivo ofrece una experiencia atractiva y clara para los jugadores.

Para mejorar la experiencia de usuario en caso de conexiones lentas, se ha implementado un efecto de "fetching" que indica la carga de preguntas. Durante este proceso, la tarjeta de la pregunta y los botones de respuesta permanecen visibles, aunque sin texto. Para informar al usuario sobre el estado de carga, se incorpora un efecto de brillo que se desplaza de izquierda a derecha. Este detalle visual permite a los jugadores saber que el juego está en proceso de obtener preguntas, añadiendo un toque de anticipación.

La interactividad toma protagonismo en la vista de partida, donde las respuestas correctas e incorrectas reciben retroalimentación inmediata. Si el jugador selecciona la respuesta correcta, la respuesta se resalta en verde antes de pasar a la siguiente pregunta. En caso de una respuesta incorrecta, la respuesta correcta se muestra en verde, mientras que la opción seleccionada por el usuario se destaca en rojo. Este enfoque de respuesta dinámica contribuye a la inmersión del jugador y mejora la experiencia general del juego.

El diseño incorpora un contador de puntos en la parte superior, proporcionando a los jugadores una medida clara de su éxito. Cada respuesta correcta incrementa el contador, fomentando la participación y la competencia. Además, se ha incluido una opción de salida intuitiva; al hacer clic en el título, los usuarios pueden volver al menú principal. Para señalar esta funcionalidad, se añade un texto animado visible al hacer hover sobre el título, asegurando que los usuarios identifiquen fácilmente esta opción de retorno al menú principal.

Para garantizar una experiencia continua y sin interrupciones, la aplicación utiliza el almacenamiento local (local storage) para preservar el token de sesión. Al emplear esta solución, incluso cuando los usuarios salen de la partida o apagan sus dispositivos, el token persiste en el almacenamiento local. Esta implementación permite recuperar el token cuando los usuarios deciden volver a jugar, asegurando que reciban nuevas preguntas y evitando la repetición de las mismas. La utilización del local storage para gestionar el token proporciona una solución eficiente y transparente, mejorando la usabilidad y la retención de datos entre sesiones de juego.

En resumen, la aplicación Vue.js ofrece una experiencia de juego envolvente y visualmente atractiva, combinando elementos de diseño intuitivo con detalles interactivos que mejoran la participación del usuario.

VanJS

Hemos implentado la web con el framework VanJS, que se caracteriza por ser muy ligero.

La trabajo de desarrollo es diferente al de react ya que no contamos con una sintaxis propia, sino que está basada en javascript puro. Por ejemplo, este sería nuestro componente `Answer`:

```
const Answer = (answer) =>
  button(
    {
      class: "answer",
      onclick: () => {
        selectedAnswer.val = answer;
        checkAnswer();
      },
    },

```

```
    answer,
  );
```

Siguiendo con la implementación, nos encontramos con algún que otro incoveniente debido a la naturaleza primitiva del framework. Un matiz que ha hecho que tengamos que reescribir un componente. Abajo se muestra el componente original, con la variable reactiva `loading` que predendíamos que actualizase todo el componente, pero no ocurría.

```
const loading = van.state(0)

const Trivia = () =>
  div(
    { id: "wrapper" },
    h2("Trivia game" + (loading && " (loading...)")),
    br(),
    div(
      { id: "question-wrapper" },
      Title(title),
      Answers,
    ),
  );
```

Así que lo intentamos de otro modo. Creamos la variable del título como reactiva y la actualizábamos entera cada vez.

```
const triviaTitle = van.state("Trivia Game")

const Trivia = () =>
  div(
    { id: "wrapper" },
    h2(triviaTitle),
    br(),
    div({ id: "question-wrapper" }, Title(title), Answers),
  );

// triviaTitle.val = "Trivia Game";
// triviaTitle.val = "Trivia Game (loading...)";
```