Guía Detallada para Personalizar tu Yincana Literaria Digital

¡Bienvenido/a a la guía de personalización de tu Yincana Literaria Digital! Esta aplicación está diseñada para ser un complemento interactivo de tu libro físico. Con esta guía, podrás modificar los desafíos, pistas, y otros contenidos para que se ajusten perfectamente a tu obra.

**Requisitos Previos:**

* **Código del Proyecto:** Debes tener los archivos de la aplicación en tu computadora.
* **Node.js y npm:** Necesarios para compilar la aplicación. Si no los tienes, puedes descargarlos desde [nodejs.org](https://www.google.com/url?sa=E&q=https%3A%2F%2Fnodejs.org%2F). npm se instala automáticamente con Node.js.
* **Git:** Necesario para interactuar con GitHub. Si no lo tienes, descárgalo desde [git-scm.com](https://www.google.com/url?sa=E&q=https%3A%2F%2Fgit-scm.com%2F).
* **Cuenta de GitHub:** Donde está alojado tu proyecto y desde donde se sirve GitHub Pages.
* **Editor de Texto:** Un programa para editar los archivos de código, como VS Code (gratuito), Sublime Text, o incluso el Bloc de Notas (aunque uno con resaltado de sintaxis es mejor).

Archivos Clave para la Personalización

La mayor parte de la personalización del contenido se realiza en un solo archivo:

* **constants.ts**: Aquí se definen todos los desafíos, sus pistas, respuestas, la imagen de la pantalla de inicio, y las clases de colores de Tailwind.
* **package.json**: Contiene los comandos (scripts) que usamos para compilar la aplicación (npm run build) y para desplegarla en GitHub Pages (npm run deploy). Normalmente no necesitarás editar este archivo para cambiar el contenido.

Paso 1: Personalizar el Contenido de los Desafíos

El corazón de tu yincana son los desafíos. Cada uno tiene varias piezas de información que puedes y debes personalizar.

1. **Abre el archivo constants.ts** en tu editor de texto.
2. Busca la sección que comienza con export const DESAFIOS: Challenge[] = [ ... ];. Esta es una lista (array) de todos los desafíos de tu yincana.
3. Cada desafío es un objeto encerrado entre llaves { }, y cada propiedad dentro del objeto tiene un "nombre" y un "valor" (ej. name: 'Los Bosques Susurrantes').

**Detalle de cada propiedad de un Desafío:**

Debes modificar estas propiedades para cada uno de tus desafíos:

* id: number;
  + **Qué es:** Un número único para identificar cada desafío.
  + **Cómo personalizar:** Asígnales números secuenciales (1, 2, 3, etc.). No los repitas.
  + **Ejemplo:** id: 1,
* name: string;
  + **Qué es:** El nombre del desafío que se mostrará en la aplicación.
  + **Cómo personalizar:** Escribe el nombre de tu desafío entre comillas simples (' ') o dobles (" ").
  + **Ejemplo:** name: 'El Misterio de la Antigua Biblioteca',
* apiKey: string;
  + **Qué es:** La "clave secreta" o código único para este desafío. **Este es el texto exacto que debe contener el Código QR correspondiente a este desafío.**
  + **Cómo personalizar:** Inventa una cadena de texto única. Puede incluir letras, números, guiones. Mantenla relativamente simple pero no demasiado obvia. ¡Debe ser EXACTA!
  + **Ejemplo:** apiKey: 'BIBLIO\_CLAVE\_77X',
* bookWord: string;
  + **Qué es:** La palabra clave que el usuario debe encontrar en el libro físico después de escanear el QR (o ingresar la apiKey manualmente).
  + **Cómo personalizar:** Escribe la palabra exacta (sin importar mayúsculas/minúsculas en la validación, pero escríbela como quieras que se vea si la muestras).
  + **Ejemplo:** bookWord: 'secreto',
* points: number;
  + **Qué es:** Los puntos que otorga este desafío al completarse.
  + **Cómo personalizar:** Un número.
  + **Ejemplo:** points: 15,
* completed: boolean;
  + **Qué es:** Indica si el desafío ha sido completado.
  + **Cómo personalizar:** **Siempre déjalo en false aquí.** La aplicación se encargará de cambiarlo a true cuando el usuario complete el desafío y guardará este estado.
  + **Ejemplo:** completed: false,
* questionPrompt: string;
  + **Qué es:** La pregunta que se le hará al usuario después de validar la bookWord.
  + **Cómo personalizar:** Escribe la pregunta.
  + **Ejemplo:** questionPrompt: "¿Cuál es el nombre del guardián que protege el tomo prohibido?",
* correctAnswer: string;
  + **Qué es:** La respuesta correcta a la questionPrompt.
  + **Cómo personalizar:** Escribe la respuesta exacta (la validación no distingue mayúsculas/minúsculas).
  + **Ejemplo:** correctAnswer: 'Argus',
* successMessage: string;
  + **Qué es:** El mensaje que se muestra cuando el usuario completa el desafío correctamente.
  + **Cómo personalizar:** Un mensaje de felicitación o narrativo.
  + **Ejemplo:** successMessage: "El guardián Argus asiente con aprobación. Has demostrado tu valía.",
* answerHintInBook: string;
  + **Qué es:** Una pista que le dice al usuario dónde buscar la respuesta a la questionPrompt dentro del libro físico.
  + **Cómo personalizar:** Sé específico sobre la ubicación en el libro.
  + **Ejemplo:** answerHintInBook: "La respuesta se encuentra en el diario del bibliotecario, página 78.",
* bookWordLocationHint: string;
  + **Qué es:** Una pista que le dice al usuario dónde buscar la bookWord en el libro físico.
  + **Cómo personalizar:** Describe la ubicación de la palabra clave.
  + **Ejemplo:** bookWordLocationHint: "Busca esta palabra en el primer párrafo del capítulo 3.",
* nextQrHint: string | null;
  + **Qué es:** Una pista que le dice al usuario dónde encontrar el siguiente Código QR en el libro físico después de completar este desafío.
  + **Cómo personalizar:** Describe la ubicación del siguiente QR. Para el último desafío, usa null (sin comillas).
  + **Ejemplo para un desafío intermedio:** nextQrHint: "El siguiente símbolo está oculto en la ilustración del mapa del tesoro.",
  + **Ejemplo para el último desafío:** nextQrHint: null,
* qrCodeLocationHint: string;
  + **Qué es:** Una pista general sobre dónde está el Código QR de *este* desafío en el libro. Se muestra antes de escanear.
  + **Cómo personalizar:** Una descripción de la ubicación del QR.
  + **Ejemplo:** qrCodeLocationHint: "El QR para este desafío está en la esquina inferior de la página 25.",
* manualKeyHint: string;
  + **Qué es:** Una pista sobre dónde encontrar la apiKey (la clave secreta en formato texto) escrita en el libro, para el desbloqueo manual.
  + **Cómo personalizar:** Describe dónde está impresa esta clave textual.
  + **Ejemplo:** manualKeyHint: "La clave (ej. BIBLIO\_CLAVE\_77X) está escrita al revés al final del índice del libro.",

**Consejos al editar constants.ts:**

* **Comas:** Cada propiedad (ej. name: 'valor') debe terminar con una coma , excepto la última propiedad dentro de un objeto { }. El último objeto de desafío en la lista no necesita una coma después de su llave de cierre } si es el último elemento de la lista [].
* **Comillas:** Los textos (strings) siempre van entre comillas simples ' ' o dobles " ". Sé consistente.
* **Guarda el archivo** después de hacer tus cambios.

Paso 2: Personalizar la Imagen de la Pantalla de Inicio

1. **Abre el archivo constants.ts**.
2. Busca la línea: export const URL\_IMAGE\_HOME\_SCREEN = '...';
3. **Cómo personalizar:** Reemplaza la URL actual con la URL de la imagen que quieres usar.
   * **Recomendación:** Sube tu imagen a un servicio de hosting de imágenes (como Imgur) o a un lugar donde puedas obtener un enlace directo a ella.
   * Asegúrate de que la URL termine en .jpg, .jpeg, .png, o .webp.
   * **Ejemplo:** export const URL\_IMAGE\_HOME\_SCREEN = 'https://mihosting.com/mi-imagen-genial.jpg';
4. **Guarda el archivo**.

Paso 3: Generar los Códigos QR

Esta parte es crucial y se hace fuera del código de la aplicación, pero la información que necesitas está en tu archivo constants.ts.

1. **Contenido de CADA Código QR:**
   * Para cada desafío que definiste en constants.ts, mira su propiedad apiKey.
   * **El texto exacto que debe contener el Código QR para ese desafío es el valor de su apiKey.**
   * **Ejemplo:**
     + Si para el Desafío 1 tienes apiKey: 'BOSQUE\_SECRETO\_01', el Código QR para el Desafío 1 debe codificar el texto: BOSQUE\_SECRETO\_01.
     + Si para el Desafío 2 tienes apiKey: 'CLAVE\_LUNAR\_XYZ', el Código QR para el Desafío 2 debe codificar el texto: CLAVE\_LUNAR\_XYZ.
2. **Cómo Generar los QRs:**
   * Usa cualquier generador de Códigos QR en línea (busca "generador QR gratuito" en Google).
   * En el generador, elige la opción de crear un QR a partir de "Texto".
   * Copia la apiKey de tu archivo constants.ts y pégala en el campo de texto del generador.
   * Genera el QR y descárgalo como una imagen (PNG o JPG suelen ser buenas opciones).
   * Repite esto para CADA uno de tus desafíos, usando su apiKey respectiva.
3. **Importancia:**  
   Cuando el usuario escanea un QR con la aplicación, la app lee el texto del QR. Luego, busca en tu lista de DESAFIOS si alguna apiKey coincide con ese texto. Si hay coincidencia, desbloquea esa etapa del desafío.

Paso 4: Actualizar la Aplicación en GitHub

Una vez que hayas personalizado constants.ts y tengas listos tus Códigos QR para el libro, necesitas subir estos cambios a GitHub para que tu sitio web se actualice.

**Parte A: Subir los cambios del código fuente (a la rama main)**

Esto actualiza los "planos" de tu aplicación.

1. **Abre tu terminal o línea de comandos** (CMD, PowerShell, Git Bash, Terminal en Mac/Linux).
2. **Navega a la carpeta raíz de tu proyecto** (donde están index.html, package.json, constants.ts, etc.).

Generated bash

cd ruta/a/tu/proyecto-yincana

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Bash

1. **Prepara los archivos modificados para ser guardados por Git:**  
   Si solo modificaste constants.ts:

Generated bash

git add constants.ts

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Bash

Si modificaste otros archivos o no estás seguro, puedes añadir todos los cambios detectados:

Generated bash

git add .

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Bash

1. **Guarda estos cambios en el historial local de Git con un mensaje descriptivo:**

Generated bash

git commit -m "Actualización de contenido de desafíos y pistas"

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Bash

(Puedes cambiar el mensaje para que describa mejor tus cambios).

1. **Sube estos cambios a tu repositorio en GitHub (a la rama main):**

Generated bash

git push origin main

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Bash

**Parte B: Compilar la aplicación y desplegarla a GitHub Pages (a la rama gh-pages)**

Esto toma tu código fuente actualizado y crea la versión "en vivo" de tu sitio web.

1. **En la misma terminal, asegúrate de estar en la carpeta raíz de tu proyecto.**
2. **Ejecuta el comando para construir/compilar tu aplicación:**

Generated bash

npm run build

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Bash

* + Espera a que este comando termine. Creará una carpeta llamada dist con los archivos optimizados de tu aplicación.
  + Si ves algún error aquí, usualmente es por un error de sintaxis en constants.ts u otro archivo .ts o .tsx. Revisa los mensajes de error para encontrar la causa.

1. **Ejecuta el comando para desplegar la carpeta dist a GitHub Pages:**

Generated bash

npm run deploy

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Bash

* + Este comando toma el contenido de la carpeta dist y lo sube a una rama especial en GitHub llamada gh-pages. GitHub Pages usa esta rama para mostrar tu sitio.
  + Espera a que termine. Usualmente dice "Published" si todo fue bien

.

Paso 5: Verificar los Cambios en Línea

1. **Espera unos minutos:** GitHub Pages puede tardar unos minutos (a veces 1, a veces 5 o más) en procesar los nuevos archivos de la rama gh-pages y actualizar tu sitio web en vivo.
2. **Abre tu sitio web:** Ve a la URL de tu yincana (ej. https://TU\_USUARIO.github.io/NOMBRE\_DEL\_REPO/).
3. **Prueba exhaustivamente:**
   * Es **muy recomendable** abrir la página en una **ventana de incógnito** de tu navegador o **limpiar la caché de tu navegador** para asegurarte de que estás viendo la versión más reciente y no una guardada anteriormente.
     + Para un "refresco forzado" que ignora la caché: Ctrl + Shift + R (Windows/Linux) o Cmd + Shift + R (Mac).
   * Verifica que los nuevos nombres de desafíos, pistas, la imagen de inicio, etc., aparezcan correctamente.
   * Prueba escanear tus nuevos QRs (si ya los tienes en físico o en pantalla) o ingresando las nuevas apiKey manualmente.

Solución de Problemas Comunes (Brevemente)

* **Errores al editar constants.ts:** Revisa comas, comillas, y que la estructura de cada objeto de desafío sea correcta.
* **Errores durante npm run build:** Lee el mensaje de error en la terminal. Usualmente te dirá en qué archivo y línea está el problema. Casi siempre son errores de sintaxis (una llave olvidada, una coma mal puesta, etc.).
* **La página en línea no se actualiza:** Lo más común es la caché del navegador. Usa modo incógnito o limpia la caché. También verifica que el proceso de npm run deploy haya terminado correctamente y que la configuración de GitHub Pages en tu repositorio apunte a la rama gh-pages (y carpeta /(root)).

¡Y eso es todo! Sé que parece mucho al principio, pero una vez que lo hagas un par de veces, el proceso de actualización se volverá más familiar. Lo más importante es ser cuidadoso al editar constants.ts y al generar los Códigos QR.

¡Mucha suerte y que tu Yincana Literaria sea un gran éxito! Si tienes más preguntas en el futuro, no dudes en consultar.