Manual Técnico de Personalización del OVA

Versión: 1.0
Fecha: 08/08/2025
Autor: Cristian Felipe Ramirez Montenegro

El OVA se basa en tecnologías web estándar, combinando **HTML, CSS y JavaScript** para ofrecer una experiencia interactiva accesible desde cualquier navegador moderno. La estructura visual y el contenido se definen mediante HTML, mientras que la lógica de funcionamiento y las interacciones se gestionan con JavaScript, aprovechando la librería **P5.js** para el manejo de gráficos y animaciones en un lienzo interactivo.

La navegación por niveles y el control de preguntas y respuestas se implementan con una arquitectura modular que separa el contenido educativo de la lógica de presentación. Esto permite que el material teórico, las actividades y los elementos interactivos estén organizados en secciones independientes, que se muestran u ocultan dinámicamente según el progreso del usuario.

El diseño está pensado para que sea fácilmente ampliable, de modo que se puedan agregar nuevos retos, contenidos multimedia o elementos interactivos sin alterar el núcleo del sistema. La disposición de cada componente favorece la reutilización y la personalización, asegurando que el OVA pueda adaptarse a distintas temáticas y estilos pedagógicos.

Objetivo:

Este documento describe el procedimiento para personalizar el contenido del OVA desarrollado con la plantilla provista.

Incluye instrucciones para agregar y modificar:

- Explicaciones teóricas en HTML.
- Preguntas y respuestas en JavaScript.
- Configuración de los tableros interactivos.
- Ajustes visuales y funcionales.

1. Personalización del Contenido Teórico

Archivo: capitulo\index.html

En el **HTML**, cada sección de contenido está envuelta en un div con la clase content y un identificador único (id).

Ejemplo:

```
<div class="content" id="enunciado1">
  <!-- Aquí puedes ingresar la explicación del contenido que desees.
       Cada div representa una página diferente identificada por los
diferentes enunciados -->
</div>
```

Procedimiento para personalizar:

1. Ubicar el div correspondiente al nivel que se desea modificar.
2. Sustituir (o incluir un nuevo enunciado) el comentario por el texto, imágenes o multimedia deseada.
3. Mantener la estructura y el atributo id intactos para que el JS pueda mostrarlo correctamente.

Importante:

Los IDs deben coincidir con los que están definidos en el **contentVisibilityMap** de sketch.js **(ver sección 4).** Si se agrega un id nuevo debe también agregarse en el **contentVisibilityMap.**

2. Estructura de Archivos

El proyecto se compone de los siguientes archivos principales:

```
/mi-ova/
│
├── index.html                ← Página principal que carga el
SCORM
├── imsmanifest.xml           ← Archivo obligatorio para SCORM
├── SCORM_API_wrapper.js      ← Driver SCORM que Moodle reconoce
├── app.js                    ← Script principal de la OVA
```

```
├── pipwerks-scorm-api-wrapper.min.js (opcional si no se usa
SCORM_API_wrapper.js)
│
├── /capitulo/                ← Carpeta con contenido del capítulo
original
│   ├── index.html            ← Archivo con el contenido html
│   ├── sketch.js             ← Script con el código del contenido
├── /css/                     ← Hojas de estilo
├── /scripts/                 ← Scripts base
│   ├── chessBoard.js         ← Script del tablero interactivo
│   ├── levelNav.js           ← Script con la lógica de navegación
├── /assets/                  ← Carpeta con imágenes y recursos
│   ├── imagen 1.png
│   ├── imagen 2.png
│   ├── imagen 3.png
│
└── ... otros archivos
```

3. Personalización de Preguntas

Archivo: **capitulo\sketch.js**

Las preguntas de cada nivel se definen en el arreglo levels:

```
const levels = [
  null, // Nivel 0 - Introducción
  [
    { id: "q1", question: "Pregunta 1", correctAnswer: "1" },
    { id: "q2", question: "Pregunta 2", correctAnswer: "0" },
    { id: "q3", question: "Pregunta 3", isStatement: true },
    { id: "q4", question: "Pregunta 4", correctAnswer: "101",
notInput: true }
  ],
  ...
];
```

Parámetros:

- **id:** Identificador único de la pregunta.
- **question:** Texto de la pregunta.
- **correctAnswer:** Respuesta correcta esperada (cadena de texto).
- **isStatement:** (Opcional) Si es true, se mostrará como afirmación sin respuesta.
- **notInput:** (Opcional) Si es true, la respuesta proviene de la interacción gráfica (tablero).

Para agregar una pregunta:

1. Seleccionar el nivel correspondiente.
2. Insertar un nuevo objeto siguiendo el formato.
3. Si la pregunta es gráfica, incluir **"notInput": true.**

4. Relación entre Contenido y Niveles

Archivo: capitulo\sketch.js

La variable contentVisibilityMap define qué elementos HTML se muestran en cada nivel:

```
const contentVisibilityMap = {
  0: ['intro'],
  1: ['enunciado1', 'decimal-label'],
  2: ['enunciado2', 'explicacion-intermedia', 'decimal-label'],
  ...
};
```

Para agregar o modificar:

1. Asegurarse de que los IDs en el HTML existan.
2. Agregar o eliminar IDs en el nivel deseado.

5. Configuración de Tableros

Cada nivel puede tener un tablero de ajedrez personalizado usando la clase ChessBoard.

Ejemplo de configuración:

```
const tablero1 = new ChessBoard(
  p,
  [gui.cellColor1, gui.cellColor2], // Colores de casillas
  [0, 0],                           // Posición inicial
  gui.cellLength,                   // Tamaño de celda
  [8, 8],                           // Filas y columnas
  tableroVacio.map(fila => [...fila]), // Estado inicial
  pieceImages,                      // Imágenes de piezas
  { minR: 7, maxR: 7, minC: 7, maxC: 7 } // Área interactiva
);
```

Parámetros clave:

- **cellColor1** / **cellColor2**: Colores alternos del tablero.
- **cellLength**: Tamaño de cada casilla (en píxeles).
- **minR** / **maxR** / **minC** / **maxC**: Límites de la zona interactiva.

6. Ajustes Visuales

Si se desea incluir un nuevo elemento interactivo (por ejemplo, un minijuego, simulador o animación), se debe:

1. Crear el nuevo componente utilizando **P5.js**, **JavaScript puro** o la librería deseada.
2. Incluir el script correspondiente en el HTML, asegurándose de cargarlo después de las librerías base.
3. Crear un nuevo div en el HTML con un id único para alojar el elemento.
4. Definir en **contentVisibilityMap** el nivel en el que este elemento debe mostrarse.

5. Si el elemento requiere capturar respuestas o estados, adaptar la lógica de **quizNavigator** para que registre dichas interacciones.

7. Ajustes Visuales

Dentro de sketch.js, la variable gui permite modificar colores y tamaños:

```
let gui = {
  "background": "#ffffff",
  "cellColor1": "#7689a0",
  "cellColor2": "#e7e8f3",
  "cellLength": 50,
}
```

Cambiar los valores según las necesidades.

8. Recomendaciones

- Mantener una copia de seguridad antes de modificar.
- Respetar la estructura de IDs y arrays para evitar errores.
- Comprobar que todas las imágenes estén en la carpeta assets con las rutas correctas.
- Validar las respuestas correctas antes de publicar.