***DISEÑO DE SOFTWARE***

***Mejora Educativa CIVE Plataforma de Apoyo al Aprendizaje***

***T C X***



***ISO/IEC 29110-4-1:2011***

***HISTORIAL DE VERSIONES***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***VERSIÓN*** | ***FECHA VIGENCIA*** | ***DETALLE DEL CAMBIO*** | ***SECCIÓN CAMBIADA*** | ***AUTOR*** | ***FECHA AUTORIZACIÓN*** |
| 1.0.0 | 09/07/2025 | Creación del documento | Todas | Equipo de desarrollo | 09/07/2025 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

***DISEÑO DE SOFTWARE***

**Identificar los componentes de software requeridos:**

* Los componentes de software requeridos en este proyecto (React + Vite + Tailwind) son:

|  |  |
| --- | --- |
| Componente | Descripción |
| App.jsx, main.jsx, UnidadX.jsx | Componentes de React que definen la estructura y navegación del sitio. |
| src/css/, tailwind.config.js | Archivos y configuración de estilos visuales (Tailwind). |
| vite.config.js | Configuración del entorno de desarrollo y despliegue. |
| img/, assets/ | Recursos gráficos utilizados en el sistema. |
| tipografia/ | Fuentes personalizadas integradas en la interfaz. |
| public/, dist/ | Recursos estáticos y ejecutables listos para producción. |

**Características de desempeño de software**

* Carga rápida mediante Vite como empaquetador.
* Renderizado eficiente con React (DOM virtual).
* Minificación de archivos en producción.
* Tailwind CSS genera clases a demanda, reduciendo el tamaño del CSS.
* Despliegue estático rápido y confiable.

**Identifica la relación entre los componentes de software**

* Relaciones clave entre los componentes del sistema:

|  |  |
| --- | --- |
| Componente | Relación |
| App.jsx | Componente principal que importa subcomponentes. |
| main.jsx | Punto de entrada que renderiza App.jsx. |
| UnidadX.jsx | Subcomponentes funcionales del sistema. |
| tailwind.config.js | Define configuración de estilos para todo el sistema. |
| vite.config.js | Afecta empaquetado y comportamiento general del sistema. |
| assets/, img/ | Recursos utilizados dentro de los componentes. |

**Características de seguridad**

* Control de acceso básico (si aplica) mediante validaciones.
* Evitar ataques XSS gracias al escape automático de React.
* Manejo adecuado de rutas y archivos en producción.
* Buenas prácticas para ocultar configuraciones sensibles.
* Políticas de seguridad de contenido gestionadas por configuración del servidor o Vite.

**Requisito de diseño de base de datos**

* En caso de integrar una base de datos, se requeriría un diseño con al menos las siguientes tablas:
  + Usuario (id, nombre, correo, rol)
  + Unidad (id, nombre, descripción)
  + Material (id, tipo, URL, unidad\_id)

Se puede implementar con servicios como Firebase o bases de datos relacionales tradicionales (MySQL, PostgreSQL).

**Manejo de errores y atributos de recuperación**

* Uso de bloques try...catch para errores en llamadas asincrónicas.
* Componentes de React con fallback para errores (Error Boundaries).
* Mensajes amigables al usuario al fallar una operación.
* Logs en consola o integración con herramientas como Sentry.
* Recuperación mediante recarga de componentes o reintento automático.

**DISEÑO DETALLADO (DE BAJO NIVEL).**

**Proporciona diseño detallado Diagrama**

**Inicio**

**|**

**v**

**Carga del componente App**

**|**

**v**

**Inicializa estados:**

**- openIndex (acordeón)**

**- menuOpen (menú hamburguesa)**

**- submenuOpen (submenú, no usado)**

**- pdfOpen (visor PDF)**

**|**

**v**

**Ejecuta useEffect para:**

**- Ajustar altura del iframe padre (si aplica)**

**- Cerrar menú si la pantalla es mayor a 768px**

**|**

**v**

**Renderiza <Router>**

**|**

**v**

**Renderiza barra de navegación:**

**- Botón menú hamburguesa**

**- Overlay para cerrar menú**

**- Lista de opciones:**

**- Bienvenida (/)**

**- Dinámica de trabajo (/dinamica)**

**- Unidades (submenu):**

**- Unidad 1 (/nueva)**

**- Unidad 2 (/unidad2)**

**- Unidad 3 (/unidad3)**

**|**

**v**

**Usuario interactúa con el menú:**

**- ¿Abre/cierra menú hamburguesa?**

**- ¿Selecciona una opción?**

**|**

**v**

**React Router determina la ruta:**

**|**

**|--- Si ruta es "/":**

**| |**

**| v**

**| Renderiza:**

**| - Sección video bienvenida (iframe YouTube)**

**| - Sección introducción (imagen + texto)**

**| - Sección equipo académico (acordeón con 3 grupos y usuarios)**

**| - Sección unidades de aprendizaje (imagen + texto)**

**| - Sección tabla de programa (<TableProgram />)**

**|**

**|--- Si ruta es "/nueva":**

**| |**

**| v**

**| Renderiza <NuevaPagina />**

**|**

**|--- Si ruta es "/dinamica":**

**| |**

**| v**

**| Renderiza <DinamicaTrabajo />**

**|**

**|--- Si ruta es "/unidad2":**

**| |**

**| v**

**| Renderiza <Unidad2 />**

**|**

**|--- Si ruta es "/unidad3":**

**| |**

**| v**

**| Renderiza <Unidad3 />**

**|**

**|--- Si ruta no existe ("\*"):**

**| |**

**| v**

**| Renderiza:**

**| - Sección video bienvenida (iframe YouTube)**

**| - Sección introducción (imagen + texto)**

**| - Sección equipo académico (acordeón)**

**| - Sección unidades de aprendizaje (imagen + texto)**

**| - Sección tabla de programa (<TableProgram />)**

**| - Sección visor PDF:**

**| - Botón para abrir PDF (ventana emergente o modal)**

**| - Enlace para descargar PDF**

**| - Si pdfOpen: muestra modal con iframe del PDF y botón cerrar**

**| - Footer con créditos**

**|**

**v**

**Fin del renderizado**

**Proporciona el formato de entrada/salida de los datos**

|  |  |  |  |
| --- | --- | --- | --- |
| **Origen** | **Estado/Variable** | **Formato** | **Destino / Acción** |
| YouTube iframe | ---- | URL string ( src ) | Render en <iframe> |
| Hook useState | openIndex: number | null | Índice active o null | Controla <details open> |
| Hook useState | menuOpen. Boolean | Truer / false | CSS classes { .menu.show, .overlay } |
| Const accordions : Array<\_> | Array de { title, users[] } | JSON literal con title:string, users: {name, img} | Mapeo o tarjetas de usuario |

**Proporciona especificaciones de las necesidades de almacenamiento de los datos**

* El PDF (programa\_de\_estudios (1).pdf) se importa como módulo y se almacena bajo /src/.
* El array sccordions vive en memoria; si fuera dinámico, podría provenir de un JSON externo o API y almacenarse en localStorage o en un backend

**Establece convenciones de denominación de los datos requeridos**

* Componentes: PascalCase (App, TableProgram, NuevaPagina).
* Archivos: .jsx con el mismo nombre de componente; CSS en /css/ con kebab-case o camelCase (app.css, dinamica.css, NuevaPagina.css).
* Variables React: camelCase (openIndex, menuOpen, toggleAccordion).
* Constantes estáticas: nombres descriptivos (accordions).

acordeon{

const accordions = [

{

title: string, // Nombre del grupo (ej. 'Especialistas en contenido')

users: [

{

name: string, // Nombre del usuario (ej. 'Ana Gómez')

img: string // URL de la imagen del usuario

},

// ...más usuarios

]

},

// ...más grupos

]

**Define el formato de las estructuras de datos requeridas**

**control de interfaz**

const [openIndex, setOpenIndex] = useState(null); // Índice del acordeón abierto

const [menuOpen, setMenuOpen] = useState(false); // Menú hamburguesa abierto/cerrado

const [submenuOpen, setSubmenuOpen] = useState(false); // Submenú (no usado)

const [pdfOpen, setPdfOpen] = useState(false); // Visor PDF abierto/cerrado

**enrutamiento**

v<Route path="/" element={ /\* Componente principal con secciones \*/ } />

<Route path="/nueva" element={<NuevaPagina />} />

<Route path="/dinamica" element={<DinamicaTrabajo />} />

<Route path="/unidad2" element={<Unidad2 />} />

<Route path="/unidad3" element={<Unidad3 />} />

<Route path="" element={ / Componente de ruta no encontrada con visor PDF \*/ } />

**Define los campos de datos y el propósito de cada elemento de datos requeridos**

Estados React

openIndex

Propósito: Controla qué acordeón está abierto en la sección de equipo académico.

Tipo: number | null

menuOpen

Propósito: Indica si el menú hamburguesa está abierto o cerrado.

Tipo: boolean

submenuOpen

Propósito: Indica si el submenú está abierto o cerrado (no se usa actualmente).

Tipo: boolean

pdfOpen

Propósito: Indica si el visor de PDF está abierto en pantalla emergente/modal.

Tipo: boolean

2. Datos de los acordeones (equipo académico)

accordions

Propósito: Lista de grupos del equipo académico, cada uno con sus usuarios.

Tipo:

Campos internos:

title: Nombre del grupo (ejemplo: 'Especialistas en contenido')

users: Array de usuarios

name: Nombre del usuario (ejemplo: 'Ana Gómez')

img: URL de la imagen del usuario

3. Funciones

toggleAccordion(index)

Propósito: Alterna el acordeón abierto/cerrado según el índice recibido.

4. Rutas y componentes

Router, Routes, Route, Link

Propósito: Navegación entre páginas/componentes de la aplicación.

TableProgram, NuevaPagina, DinamicaTrabajo, Unidad2, Unidad3

Propósito: Componentes que se renderizan según la ruta seleccionada.

5. Otros datos

programaEstudios

Propósito: Importa el archivo PDF para el visor de programa de estudios.

6. Elementos visuales y props

iframe (YouTube, PDF)

Propósito: Mostrar video de bienvenida y visor de PDF.

img (varias secciones)

Propósito: Mostrar imágenes de introducción, equipo académico y unidades.

footer

Propósito: Mostrar créditos institucionales.

**Proporciona las especificaciones de la estructura del programa**

1. Importaciones

Importa React y hooks (useState, useEffect).

Importa estilos CSS.

Importa componentes internos (TableProgram, NuevaPagina, DinamicaTrabajo, Unidad2, Unidad3).

Importa herramientas de enrutamiento de React Router (Router, Routes, Route, Link).

Importa un archivo PDF.

2. Estados principales

openIndex: Controla qué acordeón está abierto.

menuOpen: Controla la visibilidad del menú hamburguesa.

submenuOpen: Controla la visibilidad del submenú (no usado).

pdfOpen: Controla la visibilidad del visor PDF.

3. Datos

accordions: Array de objetos que representa los grupos del equipo académico y sus miembros.

4. Efectos

Ajusta la altura del iframe padre (si aplica) y actualiza en cada resize.

Cierra el menú hamburguesa si la pantalla es mayor a 768px.

5. Funciones

toggleAccordion(index): Alterna el acordeón abierto/cerrado.

6. Renderizado principal

Utiliza <Router> para manejar la navegación.

Renderiza una barra de navegación con menú hamburguesa y submenú de unidades.

Renderiza rutas con <Routes> y <Route>:

/: Página principal con video, introducción, acordeón de equipo académico, unidades y tabla de programa.

/nueva, /dinamica, /unidad2, /unidad3: Renderizan componentes específicos.

\*: Ruta comodín que muestra la página principal y el visor PDF.

7. Componentes visuales

Video de bienvenida (iframe YouTube).

Imagen de introducción.

Acordeón de equipo académico.

Imagen de unidades de aprendizaje.

Tabla de programa.

Visor PDF (modal o popup).

Footer institucional.

8. Interacción

Menú hamburguesa y submenú para navegación.

Acordeón interactivo para mostrar miembros del equipo académico.

Botón para abrir visor PDF.

Enlace para descargar PDF.

Juan Pérez María Gómez

.

Jefe de Departamento Diseñador Informático 1

Luis Rodríguez Ana Castillo

.

Diseñador 2 Diseñadora Grafica

Sofia Ramírez

.

Diseñadora Institucional