***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 | 09/07/2025 | Creación del documento | Todas | Arturo Ortiz | 09/07/2025 |
| 1.0 | 20/07/2025 | Modificación del Docto. | funcionabilidad | Arturo Ortiz | 29/07/2025 |

***DISEÑO DE SOFTWARE***

**Identificar los componentes de software requeridos:**

* Los servicios que ofrecen Microsoft Azure son:

|  |  |
| --- | --- |
| Componente | Descripción |
| App.jsx, main.jsx, UnidadX.jsx, Unidad II Dinamica de trabajo, Unidad III, nueva pagina. | Componentes de React que definen la estructura y navegación del sitio, y renderizan las distintas páginas o secciones de contenido. |
| src/css/ | Archivos de estilo, utilizados para generar clases utilitarias dinámicamente. |
| vite.config.js | Configuración del entorno de desarrollo y empaquetado del proyecto con Vite, incluyendo rutas base y optimización. |
| img/, assets/, public/ | Recursos gráficos, fuentes y archivos estáticos que se utilizan en la interfaz de usuario. |
| dist/ | Versión optimizada del sitio web, generada automáticamente con npm run build, lista para ser desplegada. |
| package.json, package-lock.json | Declaración de dependencias, scripts, configuración del proyecto y bloqueo de versiones para entornos estables. |
| .gitignore, .eslintrc.js | Archivos de configuración de control de versiones (Git) y normas de calidad del código fuente. |
| README.md, programa\_de\_estudios.pdf | Documentación del proyecto, guía de uso e información educativa como el programa de estudios. |
| Azure App Service | Servicio en la nube para hospedar el frontend React de manera escalable y segura. |
| Azure DevOps | Herramienta para gestionar la integración y entrega continua (CI/CD), automatizando pruebas y despliegue del sistema. |
| Azure Active Directory | Gestión de autenticación de usuarios internos o externos con control de permisos. |
| Azure Monitor / Application Insights | Herramientas de monitoreo y análisis para detectar errores, analizar tráfico y mejorar el desempeño del sistema en producción. |

**Características de desempeño de software**

Además de las ya existentes (como Vite y React), se recomienda agregar:

* **Escalabilidad en la nube:** El sistema puede adaptarse automáticamente al número de usuarios mediante servicios como Azure App Service, lo que evita caídas o lentitud en periodos de alta demanda.
* **Automatización del rendimiento:** El sistema puede beneficiarse del monitoreo con Azure Monitor, que permite detectar cuellos de botella y mejorar la experiencia del usuario.

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

* Relaciones clave entre los componentes de software:

|  |  |
| --- | --- |
| Componente | Relación |
| App.jsx | Componente raíz de React. Importa y organiza otros subcomponentes como UnidadX.jsx, DinamicaTrabajo, NuevaPagina, etc. |
| main.jsx | Punto de entrada de la aplicación. Renderiza <App /> dentro del DOM. |
| UnidadX.jsx | Subcomponentes que son llamados por App.jsx según la ruta activa (React Router). |
| tailwind.config.js | Configura estilos personalizados usados por los archivos .jsx y App.css. |
| vite.config.js | Define cómo se compila y sirve el proyecto. Afecta directamente el comportamiento del build (dist/) y la estructura de rutas. |
| assets/, img/, tipografia/ | Recursos visuales usados dentro de los componentes React como imágenes, íconos, fuentes. Se cargan y referencian desde los .jsx. |
| public/ | Contiene archivos estáticos accesibles directamente, como index.html. También puede incluir logos o el favicon. |
| dist/ | Resultado final del empaquetado. Este directorio se sube al servidor de producción (por ejemplo, Azure App Service). |
| Azure App Service | Servicio que aloja el contenido de dist/. Ejecuta la versión de producción del sistema React. |
| Azure SQL Database | Se relaciona con componentes de React si en el futuro el frontend consume datos dinámicos a través de una API o servicio backend. |
| Azure DevOps | Se conecta al repositorio del código (src/, package.json, etc.) para automatizar compilación y despliegue continuo. |
| Azure Monitor | Monitorea errores de producción en los componentes (App.jsx, rutas, visor PDF, etc.). Permite alertas y mejoras de rendimiento. |
| Azure Active Directory | (Opcional) Relacionado con el sistema de autenticación de usuarios que acceden a secciones protegidas del sistema. |

**Características de seguridad**

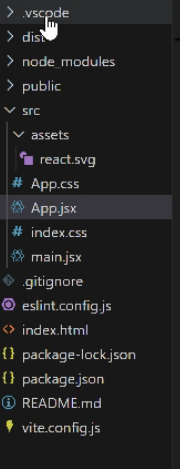
* Seguridad en la nube (Azure): El sistema puede utilizar servicios como Azure Active Directory para gestionar el acceso, y Azure Key Vault para proteger las claves, tokens y credenciales utilizadas en el frontend.
* Cumplimiento normativo: Azure cumple con estándares como ISO 27001, GDPR, y protección de datos personales aplicables en México.
* Cifrado de datos: Azure ofrece cifrado en tránsito y en reposo para proteger archivos sensibles como programas de estudio o datos de usuarios.

**Manejo de errores y atributos de recuperación**

* **Registro en la nube:** Integración con Azure Monitor y Application Insights permite capturar errores, métricas de uso, y comportamiento de usuarios en tiempo real.
* **Alertas automáticas:** Se pueden configurar alertas para notificar a los desarrolladores en caso de caídas, errores en APIs o lentitud en el sistema.
* **Integración con Sentry o herramientas similares:** mediante Azure DevOps o pipelines personalizados.

**DISEÑO DETALLADO (DE BAJO NIVEL).**

**Proporciona diseño detallado Diagrama**



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

|  |  |  |  |
| --- | --- | --- | --- |
| **Origen** | **Estado/Variable** | **Formato** | **Destino / Acción** |
| Azure App Service | process.env.\* | Variables de entorno | Acceso a configuraciones seguras en producción |
| Azure Blob Storage | url\_archivo\_pdf | string (URL pública) | Renderización en visor PDF o descarga directa |
| Azure Monitor | Error logs | JSON estructurado | Envío de métricas a Azure Application Insights |

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

* Azure ofrece almacenamiento redundante y cifrado para archivos educativos.
* Las rutas de PDF, imágenes y otros recursos pueden almacenarse directamente en Azure Blob Storage, y cargarse desde el frontend mediante URLs públicas protegidas o firmadas.
* Ideal para almacenar: videos de bienvenida, programas académicos, imágenes de presentación y posibles respaldos automáticos del contenido generado.

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

* Variables de entorno relacionadas con Azure deben tener prefijos como AZURE\_ (ej. AZURE\_STORAGE\_KEY, AZURE\_SQL\_URL).
* Archivos de configuración para despliegue en Azure pueden almacenarse en /deploy/azure/ o /azure-config/.

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

|  |  |  |
| --- | --- | --- |
| **Nombre de la estructura** | **Formato (Tipo / Sintaxis)** | **Descripción / Uso** |
| accordions | Array<Object> | Contiene los grupos del equipo académico y los usuarios que pertenecen a cada grupo. Se usa para el acordeón en la página principal. |
| accordions[i] | { title: string, users: Array<User> } | Cada grupo del acordeón tiene un título y una lista de usuarios. |
| User | { name: string, img: string } | Nombre del usuario e imagen correspondiente para mostrar en la interfaz. |
| programaEstudios | PDF File (vía import) | Archivo PDF importado en el código, mostrado mediante iframe en visor modal. |
| videoBienvenida | string (YouTube embed URL) | URL del video mostrado en la página de inicio. |
| UserLoginData | { username: string, password: string } | Estructura para autenticación de usuarios si se incorpora Azure Active Directory u otro backend. |
| API-Response | JSON | Formato estándar para recibir o enviar datos a un backend REST o API en Azure. Ej: lista de materiales, usuarios, unidades, etc. |

Estas estructuras permiten mantener el sistema organizado, flexible y escalable. Además, son compatibles con el uso de almacenamiento dinámico (como APIs REST, bases de datos en Azure o localStorage).

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

|  |  |  |
| --- | --- | --- |
| **Campo de datos** | **Tipo** | **Propósito** |
| accordions | Array<Object> | Arreglo con los grupos académicos y sus integrantes. |
| accordions[i].title | string | Título del grupo del equipo académico (ej. “Diseñadores Gráficos”). |
| accordions[i].users | Array<User> | Lista de usuarios (nombre e imagen) dentro del grupo correspondiente. |
| User.name | string | Nombre de la persona mostrada en el acordeón (ej. "Ana Gómez"). |
| User.img | string (URL) | Ruta a la imagen del usuario. Cargada desde /img/, /assets/ o incluso desde Azure Blob Storage. |
| programaEstudios | PDF importado | Archivo PDF mostrado en visor. Contiene el plan de estudios o guía educativa. |
| UserLoginData | { username, password } | Estructura básica para autenticación si se implementa login con Azure AD u otra API. |
| API-Response | JSON | Respuesta de un servicio REST, útil para cargar datos dinámicamente desde Azure o backend propio. |

Todos estos datos están pensados para vivir en estado de React (useState) o como propiedades (props) entre componentes.

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

**1. Estructura de carpetas del proyecto**

| **Carpeta / Archivo** | **Contenido / Propósito** |
| --- | --- |
| /src/ | Código fuente principal de la aplicación. Contiene componentes React, estilos y recursos. |
| /src/App.jsx | Componente raíz del sistema. Define rutas y organiza el contenido de cada página. |
| /src/main.jsx | Punto de entrada de React. Renderiza <App /> y conecta la app al index.html. |
| /src/UnidadX.jsx | Componentes funcionales para mostrar contenidos específicos (Unidad 1, Unidad 2, etc.). |
| /src/css/ | Estilos personalizados o adicionales al uso de Tailwind CSS. |
| /src/img/, /assets/ | Recursos visuales (íconos, banners, logos, etc.). |
| /src/tipografia/ poppins | Fuentes personalizadas del sistema. |
| /src/visores/ | Contiene lógica o recursos relacionados con visores PDF, videos u otros embebidos. |
| /public/ | Archivos públicos como index.html, favicon y meta etiquetas. |
| /dist/ | Carpeta generada por vite build, contiene archivos minificados para despliegue. |

**2. Componentes principales**

* **App.jsx:** Enrutador principal que organiza las rutas y secciones del sitio.
* **main.jsx:** Punto de montaje de la app en el DOM.
* **NuevaPagina, DinamicaTrabajo, Unidad2, Unidad3:** Componentes que representan páginas según rutas.
* **TableProgram:** Muestra programas educativos en formato de tabla.
* **PDFViewer, iframe:** Se usan para mostrar recursos embebidos como PDFs y videos de bienvenida.

**3. Configuraciones importantes**

| **Archivo** | **Función** |
| --- | --- |
| **vite.config.js** | **Configura la compilación, rutas base, alias, y opciones de build.** |
| **tailwind.config.js** | **Define las clases utilitarias de Tailwind personalizadas.** |
| **.eslintrc.js (opcional)** | **Reglas de estilo y calidad del código.** |
| **package.json** | **Lista de dependencias, scripts (dev, build, preview) y metadatos del proyecto.** |

**4. Ciclo de ejecución y renderizado**

**Inicio**

**↓**

**main.jsx → renderiza App.jsx**

**↓**

**App.jsx monta <Router>**

**↓**

**React Router dirige según la URL:**

**- "/" → Página principal (video + secciones + tabla + visor PDF)**

**- "/unidad2" → Unidad 2**

**- "/dinamica" → Dinámica de trabajo**

**- "/nueva" → Nueva Página**

**- "\*" → Página genérica (fallback con PDF y créditos)**

**5. Estado de la aplicación (React useState)**

| **Estado** | **Propósito** |
| --- | --- |
| **menuOpen** | **Controla el menú hamburguesa (visible/oculto).** |
| **openIndex** | **Índice del acordeón activo (equipo académico).** |
| **pdfOpen** | **Indica si el visor de PDF está abierto.** |
| **submenuOpen** | **Controla la visibilidad del submenú (no implementado en esta versión).** |

**6. Enrutamiento (React Router)**

**<Routes>**

**<Route path="/" element={<HomePage />} />**

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

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

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

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

**<Route path="\*" element={<FallbackPage />} />**

**</Routes>**

**7. Despliegue en la nube (Microsoft Azure)**

* La carpeta /dist/ es subida a Azure App Service como sitio estático.
* Configuraciones se gestionan mediante variables de entorno.
* Despliegue automatizado vía Azure DevOps Pipelines.

Juan Pérez María Gómez

.

Jefe de Departamento Diseñador Informático 1

Luis Rodríguez Ana Castillo

.

Diseñador 2 Diseñadora Grafica

Sofia Ramírez

.

Diseñadora Institucional