

# Índice General

1 Introducción 1

2 Objetivos 2

3 Diseño y Desarrollo 3

5.2 Diseño y Desarrollo Interfaz pantalla principal. 11

5.3 Diseño y Desarrollo Inicio Sesión. 11

4.Herramientas de Desarrollo 16

6.1 Entorno de Desarrollo 16

6.1.1 Visual Studio Code 16

6.1.2 React Vite 16

7 Lenguajes de Programación 16

7.1 Javascript 16

# 

# 

# 

**Herramientas de Desarrollo**

**4.1 Entorno de Desarrollo**

**4.1.1 Visual Studio Code**

El sistema se desarrollará utilizando Visual Studio Code como entorno de desarrollo integrado (IDE) principal. Visual Studio Code proporcionará las herramientas necesarias para la escritura, depuración y gestión del código fuente del sistema.

**4.1.2 React**

**Lenguajes de Programación**

**7.1 JavaScript**

JavaScript es un lenguaje de programación utilizado en el desarrollo web para crear páginas web interactivas y dinámicas. Es un lenguaje de scripting del lado del cliente, lo que significa que se ejecuta en el navegador web del usuario en lugar de en el servidor. JavaScript se utiliza para agregar interactividad, validar formularios, crear efectos visuales y mucho más en una página web.

En nuestro proyecto, JavaScript se utilizará principalmente en el frontend para mejorar la experiencia del usuario y hacer que la interfaz de usuario sea más interactiva y dinámica.

**Documentación del Componente BotonIngresar**

**Archivo:** BotonIngresar.jsx

#### Descripción General

El componente BotonIngresar es un componente funcional de React que renderiza un botón estilizado con la clase CSS botonIngresar. Este botón es utilizado para acciones de inicio de sesión en la aplicación. El componente acepta una propiedad type que define el tipo de botón, como submit, button, o reset, dependiendo del contexto en el que se utilice.

#### Importaciones

javascript

import React from 'react';

import './BotonIngresar.css';

* **React:** Importa la biblioteca de React para crear componentes funcionales.
* **BotonIngresar.css:** Importa los estilos asociados al botón desde un archivo CSS específico (BotonIngresar.css).

#### Definición del Componente

export default function BotonIngresar({ type }) {

return (

<button className='botonIngresar' type={type}>

Ingresar

</button>

);

}

##### Props

* **type (string):**
  + **Descripción:** Define el tipo de botón (submit, button, reset).
  + **Uso:** Esta propiedad es opcional, pero se recomienda especificarla para asegurar que el botón tenga el comportamiento esperado según el formulario o la acción.
  + **Ejemplo de uso:** <BotonIngresar type="submit" />

##### Estructura

* **<button>:**
  + Renderiza un botón HTML básico.
  + **Clase CSS:** Se aplica la clase botonIngresar, que define el estilo visual del botón.
  + **Texto:** El texto dentro del botón es "Ingresar", lo que indica su funcionalidad al usuario.

##### Exportación

* **export default function BotonIngresar:**
  + Exporta el componente como la exportación predeterminada, lo que permite importarlo directamente en otros archivos dentro del proyecto.

### Notas Adicionales

* **Personalización:** El botón puede ser estilizado adicionalmente modificando el archivo CSS (BotonIngresar.css). Asegúrate de mantener la consistencia con otros elementos de la interfaz de usuario.
* **Reutilización:** Este componente es altamente reutilizable y puede ser empleado en diferentes formularios de la aplicación.
* **Accesibilidad:** Considera añadir atributos adicionales como aria-label para mejorar la accesibilidad si el botón se usa en diferentes contextos.

**Documentación de Estilos CSS para la Página de Inicio de Sesión**

**Descripción General**

Este archivo CSS define los estilos utilizados en la página de inicio de sesión, específicamente para el botón de ingreso (botonIngresar) y su contenedor (contenedorBoton). Los estilos están diseñados para asegurar una apariencia coherente, centrada y visualmente atractiva del botón, tanto en su estado normal como en su estado hover.

**Estilos Definidos**

**1. .botonIngresar**

**Propósito:** Define el estilo del botón de ingreso en la página de inicio de sesión.

**Propiedades**:

padding: 10px 5px;

Establece el espacio interno del botón, con 10px en la parte superior e inferior y 5px en los lados.

background-color: #eb3e3e;

Establece el color de fondo del botón en un tono rojo.

color: white;

Define el color del texto del botón en blanco.

border: none;

Elimina cualquier borde alrededor del botón.

border-radius: 5px;

Redondea las esquinas del botón con un radio de 5px.

cursor: pointer;

Cambia el cursor a un puntero cuando se pasa sobre el botón, indicando que es clicable.

font-size: 16px;

Establece el tamaño de la fuente en 16px.

width: 120px;

Define un ancho fijo de 120px para el botón.

text-align: center;

Asegura que el texto dentro del botón esté centrado.

margin: 0 auto;

Centra el botón horizontalmente dentro de su contenedor padre si no se usan estilos flexbox.

**2. .botonIngresar:hover**

**Propósito:** Define el estilo del botón de ingreso cuando se pasa el cursor sobre él.

**Propiedades:**

background-color: #d61313;

Cambia el color de fondo del botón a un rojo más oscuro cuando el usuario pasa el cursor sobre él, proporcionando una indicación visual del estado de hover.

**3. .contenedorBoton**

**Propósito:** Define el estilo del contenedor que aloja el botón de ingreso.

**Propiedades:**

display: flex;

Aplica un modelo de diseño flexible al contenedor.

justify-content: center;

Centra el botón horizontalmente dentro del contenedor.

align-items: center;

Centra el botón verticalmente si el contenedor tiene una altura especificada.

width: 100%;

Asegura que el contenedor ocupa todo el ancho disponible en su elemento padre.

Notas Adicionales

Los colores de fondo y de texto pueden ser ajustados para que coincidan con la paleta de colores de la marca o del diseño general de la página.

Se recomienda probar la responsividad del botón y su contenedor en diferentes dispositivos y tamaños de pantalla para asegurar que la experiencia del usuario sea consistente.

### Documentación del Componente Header

**Archivo:** Header.jsx

#### Descripción General

El componente Header es un componente funcional de React que renderiza el encabezado de la aplicación, específicamente para una biblioteca áulica. Este encabezado incluye un logotipo y un título, y está estilizado mediante un archivo CSS importado (header.css). El Header se utiliza para proporcionar una identidad visual coherente en la parte superior de la página.

#### Importaciones

jimport React from 'react';

import './header.css';

* **React:** Importa la biblioteca de React para crear componentes funcionales.
* **header.css:** Importa los estilos específicos asociados al encabezado desde un archivo CSS.

#### Definición del Componente

const Header = () => {

return (

<header className="header">

<div className="logo-container">

<img src="biblioteca-front\logo.png" alt="Logo" className="logo" />

</div>

<h1 className="header-title">Biblioteca Aulica</h1>

</header>

);

};

export default Header;

##### Estructura del Componente

* **<header className="header">:**
  + Contenedor principal del encabezado. Se le asigna la clase header para aplicar los estilos CSS correspondientes.
* **<div className="logo-container">:**
  + Contenedor del logotipo. La clase logo-container permite aplicar estilos específicos para el posicionamiento y la presentación del logotipo.
* **<img src="biblioteca-front\logo.png" alt="Logo" className="logo" />:**
  + **src:** Especifica la ruta a la imagen del logotipo (logo.png).
  + **alt:** Proporciona un texto alternativo "Logo" para mejorar la accesibilidad y el SEO.
  + **className="logo":** Aplica la clase CSS logo para definir los estilos del logotipo.
* **<h1 className="header-title">Biblioteca Aulica</h1>:**
  + Renderiza el título del encabezado como un h1, con el texto "Biblioteca Aulica".
  + La clase header-title se utiliza para aplicar estilos específicos al título.

##### Exportación

* **export default Header:**
  + Exporta el componente Header como la exportación predeterminada del archivo, lo que permite importarlo y usarlo en otros componentes o páginas dentro del proyecto.

### Documentación de Estilos CSS para Contenedores de Ingreso y Perfiles

#### Descripción General

Este archivo CSS define los estilos aplicados a los contenedores principales utilizados en la página de inicio de sesión, incluyendo contenedorIngresar y contenedorPerfiles. También incluye reglas de medios (@media) para hacer que estos contenedores sean responsivos y se adapten a pantallas más pequeñas.

#### Estilos Definidos

##### 1. .contenedorIngresar

* **Propósito:** Estiliza el contenedor principal utilizado para el formulario de inicio de sesión.
* **Propiedades:**
  + font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;  
    Define una lista de fuentes seguras para web que se aplican al contenido dentro del contenedor.
  + display: flex;  
    Aplica un modelo de diseño flexible al contenedor para organizar su contenido.
  + flex-direction: column;  
    Organiza los elementos hijos en una columna.
  + text-align: center;  
    Centra el texto dentro del contenedor.
  + border: 1px solid black;  
    Aplica un borde negro de 1px alrededor del contenedor.
  + border-radius: 5px;  
    Redondea las esquinas del contenedor con un radio de 5px.
  + background-color: antiquewhite;  
    Establece un color de fondo antiquewhite para el contenedor.
  + max-width: 500px;  
    Limita el ancho máximo del contenedor a 500px.
  + width: 90%;  
    Establece que el contenedor ocupe el 90% del ancho disponible en su contenedor padre.

##### 2. .contenedorPerfiles

* **Propósito:** Estiliza el contenedor utilizado para mostrar los perfiles de usuario en la página de inicio de sesión.
* **Propiedades:**
  + display: flex;  
    Aplica un modelo de diseño flexible al contenedor.
  + flex-direction: row;  
    Organiza los elementos hijos en una fila.
  + align-items: center;  
    Alinea verticalmente los elementos hijos al centro del contenedor.
  + justify-content: center;  
    Centra los elementos hijos horizontalmente dentro del contenedor.
  + flex-wrap: wrap;  
    Permite que los elementos hijos se envuelvan en múltiples filas si el espacio horizontal es limitado.

##### 3. @media (max-width: 600px)

* **Propósito:** Define reglas CSS que se aplican cuando la pantalla tiene un ancho máximo de 600px, haciendo que la página sea responsiva en dispositivos móviles.
* **Propiedades:**
  + .contenedor:
    - width: 100%;  
      Asegura que el contenedor ocupe el 100% del ancho disponible en pantallas pequeñas.
    - max-width: none;  
      Elimina la restricción de ancho máximo para permitir que el contenedor se expanda completamente en dispositivos móviles.
  + #user:
    - width: 40px;  
      Establece el ancho del elemento con id user a 40px en pantallas pequeñas.
    - height: 40px;  
      Establece la altura del elemento con id user a 40px, asegurando que el tamaño sea consistente.

### Documentación del Archivo Principal (main.js o index.js)

#### Descripción General

Este archivo es el punto de entrada principal de la aplicación React. Se encarga de inicializar y renderizar la aplicación dentro del DOM, aplicando las configuraciones necesarias para garantizar que la aplicación funcione en modo estricto, lo que ayuda a identificar posibles problemas en el código.

#### Importaciones

jimport { StrictMode } from 'react';

import { createRoot } from 'react-dom/client';

import App from './App.jsx';

import './index.css';

* **StrictMode:**
  + Importa el componente StrictMode de React, que activa verificaciones y advertencias adicionales en el desarrollo para identificar posibles problemas en la aplicación.
* **createRoot:**
  + Importa la función createRoot de react-dom/client, que se utiliza para crear el nodo raíz en el que se montará la aplicación React en el DOM.
* **App:**
  + Importa el componente principal App desde App.jsx. Este componente representa la estructura general de la aplicación y es el componente raíz que se renderiza.
* **index.css:**
  + Importa el archivo CSS global index.css, que contiene los estilos generales aplicados a toda la aplicación.

#### Renderización de la Aplicación

createRoot(document.getElementById('root')).render(

<StrictMode>

<App />

</StrictMode>,

);

* **createRoot(document.getElementById('root')):**
  + Selecciona el elemento DOM con el id root y crea un nodo raíz en el que se montará la aplicación React.
  + **document.getElementById('root'):** Se asume que existe un elemento con el id root en el archivo HTML principal (index.html) donde se insertará la aplicación.
* **.render( <StrictMode> <App /> </StrictMode> ):**
  + Monta y renderiza el componente App dentro del nodo raíz.
  + **<StrictMode>:** Envuelve el componente App en el StrictMode de React, lo que activa ciertas verificaciones adicionales durante el desarrollo para asegurar que el código esté libre de prácticas problemáticas.

##### Notas sobre StrictMode:

* StrictMode no afecta el comportamiento de la aplicación en producción; solo se activa durante el desarrollo.
* Ayuda a identificar posibles problemas de seguridad, errores de ciclo de vida de componentes, y más.

### Notas Adicionales

* **Estructura del Proyecto:** Este archivo es fundamental en cualquier proyecto React, ya que establece el vínculo entre el código React y el DOM del navegador.
* **Customización:** Si se cambia el id del elemento raíz en el HTML, se deberá actualizar document.getElementById('root') para que coincida.
* **Optimización:** Asegúrate de utilizar createRoot para aprovechar las mejoras de rendimiento y las nuevas características de React 18 en adelante.