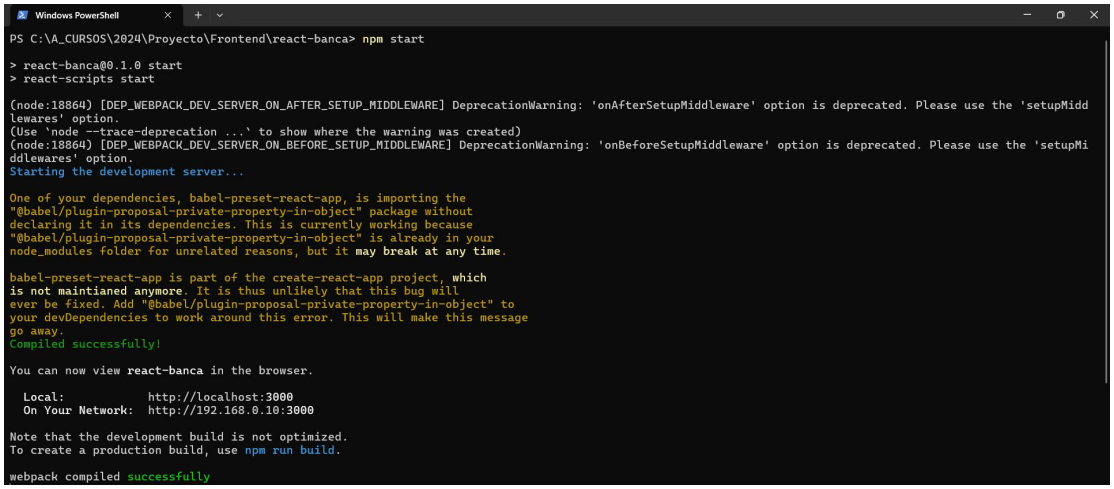
PROYECTO EJERCICIOS PRACTICOS CON REACT

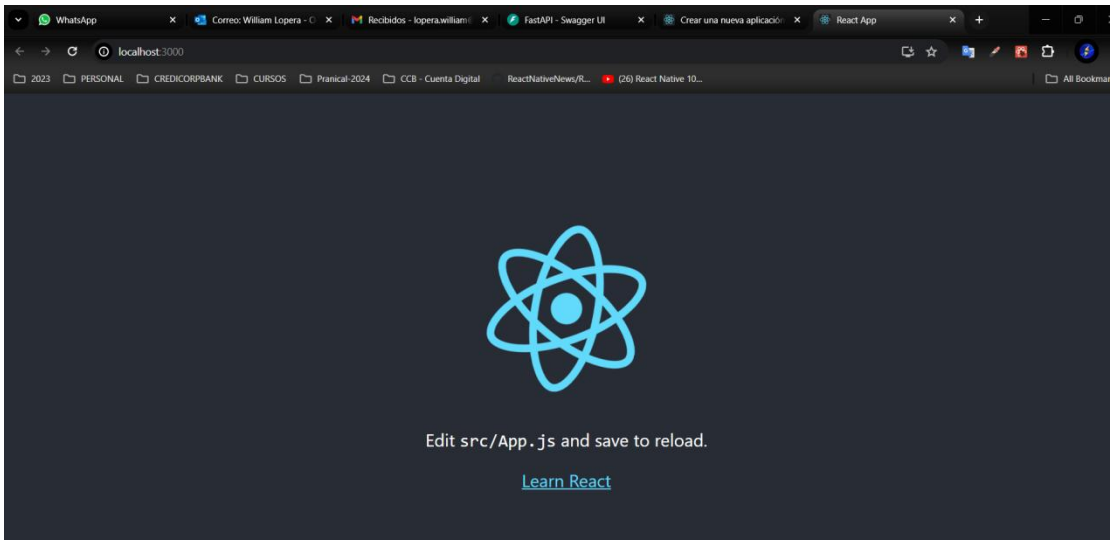
Proyecto de estudio de ejercicios para react utilizando librerías básicas de React

Se mostrara un ejercicio y su solución utilizando lo menos posible internet ni uso de aplicaciones de chatbot de inteligencia artificial . Para practicar React, Javascript, Html y CSS.

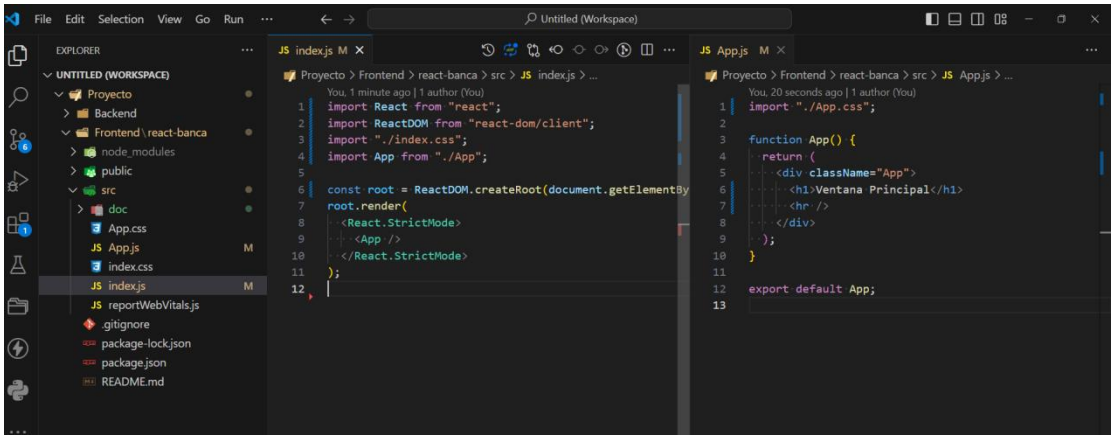
- *Crear proyecto*

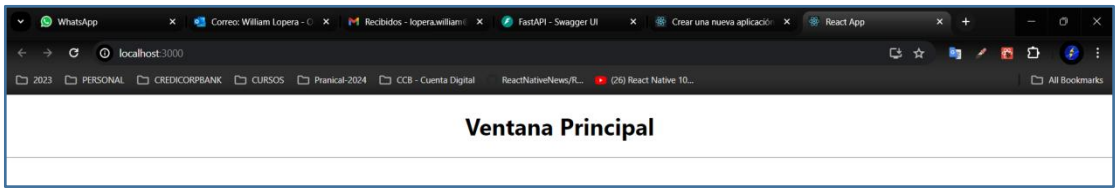
- *Subir servidor*





- VSCODE: Depurar el código





Código

Se crearon cuatro componentes importantes:

- Header: Cabecera de la Pantalla
- Footer: Mensajes de información al usuario y del sistema
- Menu: Area para agregar valores verticales de Menú (dinámico)
- Body: Cuerpo del programa. Este área varia a medida que se seleccione componentes del menú

Además se crearon componentes para el manejo de:

- mensaje: Control de mensajes
- HeaderProcess: Control de titulo, descripción y problema a resolver

Uso de mobx para Tener un almacén común de datos (por ahora mensajes al usuario)

- Librerías:

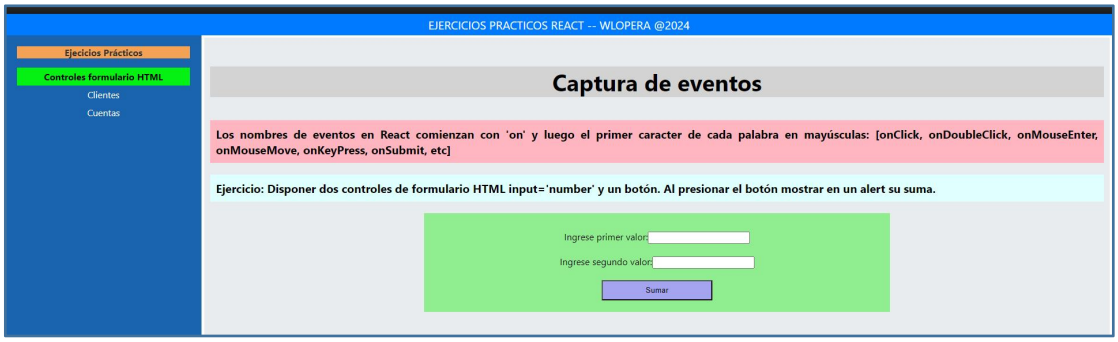
```
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.7.2",
    "mobx": "^6.12.4",
    "mobx-react": "^9.1.1",
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
"react-scripts": "5.0.1",
 "styled-components": "^6.1.11",
    "web-vitals": "^2.1.4"
```

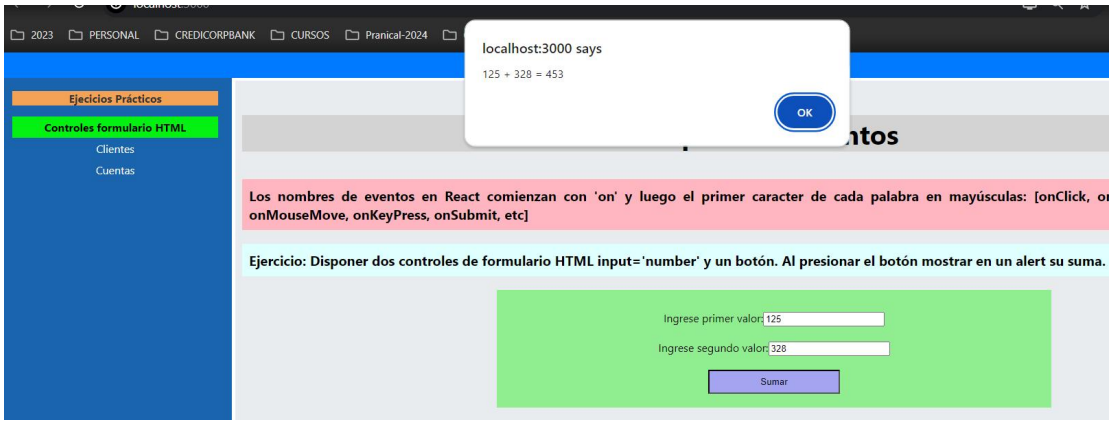
Ejercicio 1

Disponer dos controles de formulario HTML input='number' y un botón. Al presionar el botón mostrar en un alert su suma."

```
import React from "react";
import { HeaderProcess } from "../../components/headerProcess/HeaderProcess";
import styled from "styled-components";

const title = "Captura de eventos";
const description =
  "Los nombres de eventos en React comienzan con 'on' y luego el primer caracter de cada
palabra en mayúsculas: [onClick, onDoubleClick, onMouseEnter, onMouseMove, onKeyPress,
onSubmit, etc]";
const exercise =
  "Ejercicio: Disponer dos controles de formulario HTML input='number' y un botón. Al
presionar el botón mostrar en un alert su suma.";

export const EventCapture = () => {
  const sum = (e) => {
    e.preventDefault();
    const x = parseFloat(e.target.x.value);
    const y = parseFloat(e.target.y.value);
    return alert(`${x} + ${y} = ${x + y}`);
  };
  return (
    <Container>
      <HeaderProcess
        title={title}
        description={description}
        exercise={exercise}
      />

      <Form onSubmit={sum}>
        <Label>
          Ingrese primer valor:
          <input type="number" name="x" />
        </Label>
        <Label>
          Ingrese segundo valor:
          <input type="number" name="y" />
        </Label>
        <Button>Sumar</Button>
      </Form>
    </Container>
  );
};

const Container = styled.div`
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
`;

const Form = styled.form`
  display: flex;
  background-color: lightgreen;
  flex-direction: column;
```

```
  align-items: center;
  width: 40vw;
  padding: 20px;
`;

const Label = styled.label`
  padding: 10px;
`;

const Button = styled.button`
  margin-top: 10px;
  background-color: rgb(164, 164, 241);
  width: 10vw;
  height: 4vh;
`;
```



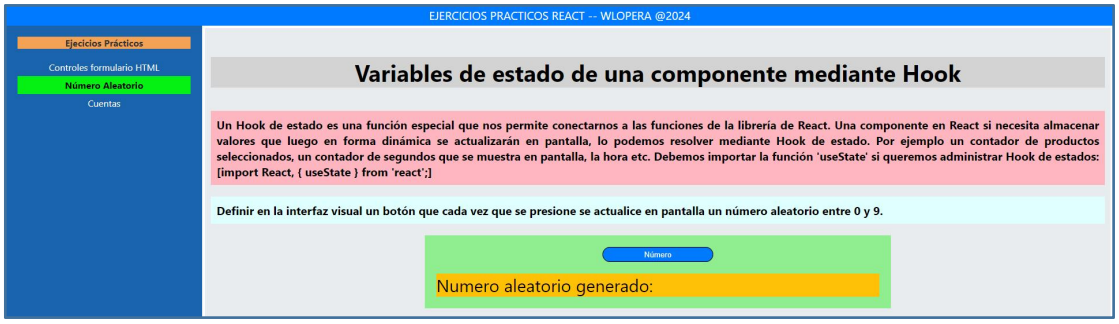


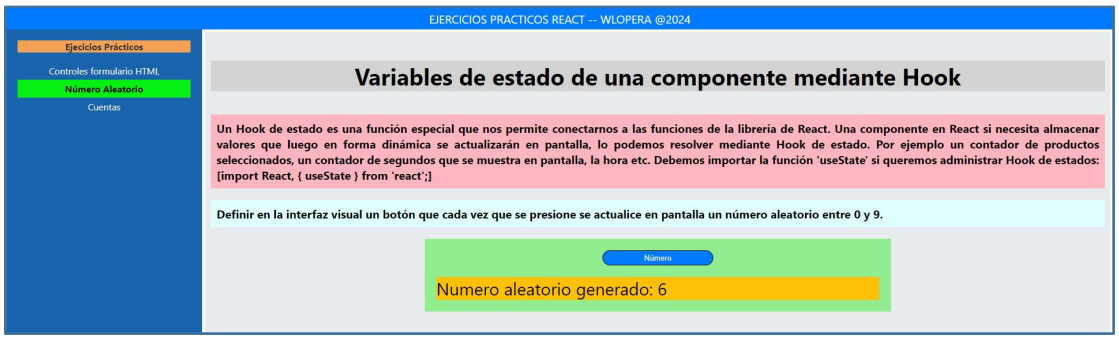
Ejercicio 2

Definir en la interfaz visual un botón que cada vez que se presione se actualice en pantalla un número aleatorio entre 0 y 9.

```
import React, { useState } from "react";
import styled from "styled-components";

import { HeaderProcess } from "../../components/headerProcess/HeaderProcess";

const title = "Variables de estado de una componente mediante Hook";
const description =
  "Un Hook de estado es una función especial que nos permite conectarnos a las funciones de la librería de React. Una componente en React si necesita almacenar valores que luego en forma dinámica se actualizarán en pantalla, lo podemos resolver mediante Hook de estado. Por ejemplo un contador de productos seleccionados, un contador de segundos que se muestra en pantalla, la hora etc. Debemos importar la función 'useState' si queremos administrar Hook de estados: [import React, { useState } from 'react';]";
const exercise =
  "Definir en la interfaz visual un botón que cada vez que se presione se actualice en pantalla un número aleatorio entre 0 y 9.";

export const RandomNumber = () => {
  const [number, setNumber] = useState();

  const handleRandomNumber = () => {
    const value = Math.floor(Math.random() * 10);
    setNumber(value);
  };

  return (
    <Container>
      <HeaderProcess
        title={title}
        description={description}
        exercise={exercise}
      />
      <Div>
        <Button onClick={handleRandomNumber}>Número</Button>
        <Label>Numero aleatorio generado: {number}</Label>
      </Div>
    </Container>
  );
};

const Container = styled.div`
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
`;

const Div = styled.div`
  display: flex;
  flex-direction: column;
  align-items: center;
  padding: 20px;
  background-color: #90ee90;
`;
```

```
const Button = styled.button`
  background-color: #007bff;
  color: white;
  border: 1px solid black;
  border-radius: 20px;
  width: 10vw;
  height: 3vh;
`;

const Label = styled.label`
  margin-top: 20px;
  font-size: 30px;
  color: black;
  width: 40vw;
  background-color: #ffc107;
  text-align: left;
`;
```





Ejercicio 3

Almacenar en el estado de la componente el siguiente vector con artículos:

```
[
        {
                codigo: 1,
                descripcion: 'papas',
                precio: 12.52
        },{
                codigo: 2,
                descripcion: 'naranjas',
                precio: 21
        },{
                codigo: 3,
                descripcion: 'peras',
                precio: 18.20
        }
]
```

- Mostrar en una tabla HTML dichos datos.
- Borrar el artículo cuando se presione el botón borrar de la tabla.

Data

```
export const ARTICLES = [
  {
    codigo: 1,
    descripcion: "papas",
    precio: 12.52,
  },
  {
    codigo: 2,
    descripcion: "naranjas",
    precio: 21,
  },
  {
    codigo: 3,
    descripcion: "peras",
    precio: 18.2,
  },
];
```

Componente

```
import React, { useState } from "react";
import styled from "styled-components";

import { HeaderProcess } from "../../components/headerProcess/HeaderProcess";
import { ARTICLES } from "../../data/Data";

const title = "Variables de estado de una componente mediante Hook";
const description =
```

```
  "Un Hook de estado es una función especial que nos permite conectarnos a las funciones de la librería de React. Una componente en React si necesita almacenar valores que luego en forma dinámica se actualizarán en pantalla, lo podemos resolver mediante Hook de estado. Por ejemplo un contador de productos seleccionados, un contador de segundos que se muestra en pantalla, la hora etc. Debemos importar la función 'useState' si queremos administrar Hook de estados: [import React, { useState } from 'react';]";
const exercise =
  "Ejercicio: Almacenar en el estado de la componente un vector con artículos. Mostrar en una tabla HTML dichos datos. Borrar el artículo cuando se presione el botón borrar de la tabla. [Se utiliza una data dummy de atículos]";

const TableArticles = () => {
  const [articles, setArticles] = useState(ARTICLES);

  const handleDeleteArticle = (codigo) => {
    const filter = articles.filter((article) => article.codigo !== codigo);
    setArticles(filter);
  };

  return (
    <Container>
      <HeaderProcess
        title={title}
        description={description}
        exercise={exercise}
      />
      <table>
        <thead>
          <tr>
            <TH>Código</TH>
            <TH>Descripción</TH>
            <TH>Precio</TH>
            <TH>Acción</TH>
          </tr>
        </thead>
        <tbody>
          {articles.map((article) => (
            <TR key={article.codigo}>
              <TD>{article.codigo}</TD>
              <TD>{article.descripcion}</TD>
              <TD>{article.precio}</TD>
              <TD onClick={() => handleDeleteArticle(article.codigo)}>x</TD>
            </TR>
          ))}
        </tbody>
      </table>
    </Container>
  );
};

export default TableArticles;

const Container = styled.div`
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
`;

const TR = styled.tr`
  border: 1px solid black;
```

```
  &:hover {
    background-color: #bdbdbd;
  }
`;

const TH = styled.td`
  border: 1px solid black;
  font-size: 20px;
  width: 10vw;
  background-color: #28a745;
  color: white;
`;

const TD = styled.td`
  border: 1px solid black;
  font-size: 16px;
  font-weight: bold;
`;
```