Apuntes para Examen

Ejercicio Agenda → Axios, Modal, CRUD

Agenda-Axios

App.jsx

```
import { useState, useEffect } from "react";
import Header from "./components/Header";
import Body from "./components/Body";
import Swal from "sweetalert2";
import ServicioContactos from "./services/servicioContactos";

function App() {

  const [contactos, setContactos] = useState([])

  useEffect(() => {
    ServicioContactos.getAll()
      .then((response) => {
        setContactos(response.data);
      })
      .catch((error) => {
        Swal.fire({
          title: "¿Tienes Internet?",
          text: "No consigo descargar los contactos",
          icon: "question",
        });
      });
  }, []);
  return (
```

```
    <>
      <Header contactos={contactos}></Header>
      <Body contactos={contactos} setContactos={setContactos}>
      </Body>

    </>
  );
}

export default App;
```

Services/http-axios.js

```
import axios from "axios";

const http = axios.create({
  baseURL: "http://localhost:3000/",
  headers: {
    "Content-Type": "application/json",
  },
});

export default http;
```

Services/servicioContactosjs

```


import http from "./http-axios.js";

class ServicioContactos {
  getAll() {
    return http.get("/contactos");
  }

  get(id) {
    return http.get(`/contactos/${id}`);
```

```
  }

  create(data) {
    return http.post("/contactos", data);
  }

  update(id, data) {
    console.log(id,data)
    return http.put(`/contactos/${id}`, data);
  }

  delete(id) {
    return http.delete(`/contactos/${id}`);
  }
}

export default new ServicioContactos();
```

Agenda-Modal

Components/Modal.jsx

```
import React from 'react';


const Modal = ({ isOpen, onClose, children }) => {
  if (!isOpen) return null;

  return (
    <div className="modal-overlay">
      <div className="modal-content">
        <button className="close-btn" onClick={onClose}>X</butt
        {children}
      </div>
```

```
    </div>
  );
};

export default Modal;
```

Components/Body.jsx → Parte modal

```
import React, { useState } from "react";
import Modal from '../components/Modal';
import Crear from './crud/Crear';
import Consultar from './crud/Consultar';
import Editar from './crud/Editar';
import ContactoBorrar from './crud/Borrar'; // Asegúrate de impo

const Body = ({ contactos, setContactos }) => {
  const [modals, setModals] = useState({
    crear: false,
    consultar: false,
    editar: false,
  });
  const [contactoSeleccionado, setContactoSeleccionado] = useSta

  const gestionarModal = (tipoModal, estadoAbierto) => {
    setModals((prevModals) => ({
      ...prevModals,
      [tipoModal]: estadoAbierto,
    }));
  };

  const consultar = (contacto) => {
    setContactoSeleccionado(contacto);
    gestionarModal("consultar", true);
  };

  const editar = (contacto) => {
```

```
    setContactoSeleccionado(contacto);
    gestionarModal("editar", true);
  };

  const crear = () => {
    gestionarModal("crear", true);
  };

  const borrar = (contacto) => {
    ContactoBorrar(contacto, contactos, setContactos); // Llamad
  };

  return (
    <div className="body">
      <ul>
        {contactos.length > 0 ? (
          contactos.map((contacto) => (
            <li key={contacto.id}>
              <div className="caja-contacto">
                <strong>{contacto.nombre}</strong>: {contacto.nu
                <button onClick={() => consultar(contacto)}>
                  Consultar
                </button>
                <button onClick={() => borrar(contacto)}>Borrar<
                <button onClick={() => editar(contacto)}>Editar<
              </div>
            </li>
          ))
        ) : (
          <p>No se encontraron contactos.</p>
        )}
      </ul>
      <button onClick={crear}>Añadir</button>

      <Modal
        isOpen={modals.crear}
```

```
          onClose={() => gestionarModal("crear", false)}
        >
          <Crear
            contactos={contactos}
            setContactos={setContactos}
            onClose={() => gestionarModal("crear", false)}
          />
        </Modal>

        <Modal
          isOpen={modals.consultar}
          onClose={() => gestionarModal("consultar", false)}
        >
          {contactoSeleccionado && (
            <Consultar contacto={contactoSeleccionado} />
          )}
        </Modal>

        <Modal
          isOpen={modals.editar}
          onClose={() => gestionarModal("editar", false)}
        >
          {contactoSeleccionado && (
            <Editar
              contacto={contactoSeleccionado}
              setContactos={setContactos}
              onClose={() => gestionarModal("editar", false)}
            />
          )}
        </Modal>
      </div>
    );
};

export default Body;
```

Agenda-CRUD → con formulario

Components/crud/borrar.js

```
import ServicioContactos from "../../services/servicioContactos
import Swal from "sweetalert2";

const ContactoBorrar = (contacto, contactos, setContactos) => {
  Swal.fire({
    title: "¿Estás seguro?",
    text: "No podrás revertir esta acción",
    icon: "warning",
    showCancelButton: true,
    confirmButtonColor: "#d33",
    cancelButtonColor: "#3085d6",
    confirmButtonText: "Sí, eliminar",
    cancelButtonText: "Cancelar",
  }).then((result) => {
    if (result.isConfirmed) {
      // Llamada al servicio para eliminar el contacto
      ServicioContactos.delete(contacto.id)
        .then(() => {
          Swal.fire("Contacto borrado correctamente");

          // Filtrar el contacto eliminado de la lista
          const nuevosContactos = contactos.filter((c) => c.id
          setContactos(nuevosContactos);

          /*
                    const eliminarProducto = (nombreProducto) =>
                                  const nuevosProductos = proc
                                  setProductosJson(nuevosProdu
  };
           */
```

```
          Swal.fire(
            "¡Eliminado!",
            "El contacto ha sido eliminado.",
            "success"
          );
        })
        .catch(() => {
          Swal.fire("ERROR, No se ha borrado el contacto");
        });
    }
  });
};

export default ContactoBorrar;
```

Components/crud/Consultar.jsx

```
import React from 'react';

const AficionConsultar = ({ contacto }) => {

  return (
    <div>
      <h2>{contacto.nombre}</h2>
      <p><strong>Descripción:</strong> {contacto.numero}</p>
    </div>
  );
};

export default AficionConsultar;
```

Components/crud/Crear.jsx

```jsx
import React, { useState } from 'react';
import Swal from 'sweetalert2';
import servicioContactos from '../../services/servicioContactos

function ContactoCrear({ contactos, setContactos, onClose }) {
  // Almacenar los errores del formulario
  const [errores, setErrores] = useState({});

  // Almacenar los valores del formulario
  const [form, setForm] = useState({
    nombre: '',
    numero: '',
  });

  ////////////////////////////////////
  // Función para gestionar los cambios en los campos del formul
  ////////////////////////////////////
  const gestionarCambio = (e) => {
    const { name, value } = e.target;

    setForm({
      ...form,
      [name]: value,
    });
  };

  ////////////////////////////////////
  // Función de validación
  ////////////////////////////////////
  const validar = () => {
    const nuevosErrores = {};

    // Validación para "nombre"
    if (!form.nombre.trim()) {
      nuevosErrores.nombre = 'El nombre es obligatorio';
```

```
    }

    // Validación para "numero"
    if (!form.numero.trim()) {
      nuevosErrores.numero = 'El número es obligatorio';
    } else if (!/^\d+$/.test(form.numero)) {
      nuevosErrores.numero = 'El número debe ser solo dígitos';
    }

    /*
        // Validación de email
        if (!form.email) {
          nuevosErrores.email = 'El email es obligatorio.';
        } else if (!/\S+@\S+\.\S+/.test(formulario.email)) {
          nuevosErrores.email = 'El email no es válido.';
    }

    */

    setErrores(nuevosErrores);

    // Retorna true si no hay errores, de lo contrario retorna f
    return Object.keys(nuevosErrores).length === 0;
  };

  // Función para manejar el envío del formulario
  const enviarFormulario = (e) => {
    e.preventDefault();

    // Validar antes de enviar
    if (validar()) {
      console.clear();
      console.log('Formulario Enviado', form);

      const nuevoContacto = {
        nombre: form.nombre,
```

```
        numero: form.numero,
      };

      // Enviar por Axios al JSON de la BD
      servicioContactos.create(nuevoContacto)
        .then(response => {
          Swal.fire("Contacto creado correctamente");

          // Limpiar el formulario después de agregar
          setForm({
            nombre: '',
            numero: '',
          });

          // Le ponemos el id correcto de la BD
          nuevoContacto.id = response.data.id;

          // Actualizar el estado local de contactos
          setContactos([...contactos, nuevoContacto]);

          // Cerramos el modal
          onClose();

        })
        .catch(error => {
          Swal.fire("ERROR, Al crear contacto");
        });
    }
  };

  return (
    <form onSubmit={enviarFormulario}>
      {/* Campo de texto para nombre */}
      <label htmlFor="nombre">Nombre del Contacto</label>
      <input
        id="nombre"
```

```
          type="text"
          name="nombre"
          value={form.nombre}
          onChange={gestionarCambio}
          placeholder="Escribe el nombre del contacto"
        />
        {errores.nombre && <p className="error">{errores.nombre}</

        {/* Campo de texto para número de contacto */}
        <label htmlFor="numero">Número de Teléfono</label>
        <input
          id="numero"
          type="text"
          name="numero"
          value={form.numero}
          onChange={gestionarCambio}
          placeholder="Escribe el número de teléfono"
        />
        {errores.numero && <p className="error">{errores.numero}</

        {/* Botón de envío */}
        <button type="submit">Enviar</button>
      </form>
    );
}

export default ContactoCrear;
```

Components/crud/Editar.jsx

```
import React, { useState } from 'react';
import servicioContactos from '../../services/servicioContactos
import Swal from 'sweetalert2';

function ContactoEditar({ contacto, setContactos, onClose }) {
  // Almacenar los errores del formulario
```

```
  const [errores, setErrores] = useState({});

  // Almacenar los valores del formulario
  const [form, setForm] = useState({
    nombre: contacto.nombre,
    numero: contacto.numero,
  });

  ////////////////////////////////////
  // Función para gestionar los cambios en los campos del formul
  ////////////////////////////////////
  const gestionarCambio = (e) => {
    const { name, value } = e.target;

    setForm({
      ...form,
      [name]: value,
    });
  };

  ////////////////////////////////////
  // Función de validación
  ////////////////////////////////////
  const validar = () => {
    const nuevosErrores = {};

    // Validación para "nombre"
    if (!form.nombre.trim()) {
      nuevosErrores.nombre = 'El nombre es obligatorio';
    }

    // Validación para "numero"
    if (!form.numero.trim()) {
      nuevosErrores.numero = 'El número es obligatorio';
    } else if (!/^\d+$/.test(form.numero)) {
      nuevosErrores.numero = 'El número debe ser solo dígitos';
```

```
    }

    setErrores(nuevosErrores);

    // Retorna true si no hay errores, de lo contrario retorna f
    return Object.keys(nuevosErrores).length === 0;
  };

  // Función para manejar el envío del formulario
  const enviarFormulario = (e) => {
    e.preventDefault();

    // Validar antes de enviar
    if (validar()) {
      console.clear();
      console.log('Formulario Enviado', form);

      const editarContacto = {
        nombre: form.nombre,
        numero: form.numero,
      };

      // Enviar por Axios al JSON de la BD
      servicioContactos.update(contacto.id, editarContacto)
        .then(response => {
          Swal.fire("Contacto Actualizado correctamente");

          // Limpiar el formulario después de agregar
          setForm({
            nombre: '',
            numero: '',
          });

          // Actualizar el estado local de contactos
          servicioContactos.getAll()
            .then((response) => {
```

```
            setContactos(response.data);
          });

        // Cerramos el modal
        onClose();

      })
      .catch(error => {
        Swal.fire("ERROR, Al actualizar el contacto");
      });
  }
};

return (
  <form onSubmit={enviarFormulario}>
    {/* Campo de texto para nombre */}
    <label htmlFor="nombre">Nombre del Contacto</label>
    <input
      id="nombre"
      type="text"
      name="nombre"
      value={form.nombre}
      onChange={gestionarCambio}
      placeholder="Escribe el nombre del contacto"
    />
    {errores.nombre && <p className="error">{errores.nombre}</

    {/* Campo de texto para número de contacto */}
    <label htmlFor="numero">Número de Teléfono</label>
    <input
      id="numero"
      type="text"
      name="numero"
      value={form.numero}
      onChange={gestionarCambio}
      placeholder="Escribe el número de teléfono"
```

```
        />
        {errores.numero && <p className="error">{errores.numero}</
        {/* Botón de envío */}
        <button type="submit">Actualizar</button>
      </form>
    );
}

export default ContactoEditar;
```

Agenda-Header → Toggle

```
import React from "react";
import { useState } from "react";
const Header = ({ contactos }) => {
  const [usuarioVisible, setUsuarioVisible] = useState(false);

  const toggleUsuario = () => {
    setUsuarioVisible(!usuarioVisible);
  };

  const nContactos = contactos.length

  return (
    <div className="header">
      <div className="n-contactos">
        <h2>Nº de contactos: {nContactos}</h2>
      </div>

      <div className="usuario" onClick={toggleUsuario}>
        <img
          src="https://img.icons8.com/?size=100&id=42384&format=
          alt="Imagen"
          width={50}
```

```
          />
          <h3>Lucas Chacon</h3>
        </div>

        {usuarioVisible && (
          <div className="toggle-usuario">
            <h4>Lucas Chacon</h4>
            <p>El rey del front</p>
            <p>{nContactos} contactos en la lista</p>
          </div>
        )}
      </div>
    );
};

export default Header;
```

Agenda-data.json

```
{
  "contactos": [
    {
      "id": "d",
      "nombre": "Natalia",
      "numero": "736653543"
    },
    {
      "id": "b9ea",
      "nombre": "pepe",
      "numero": "465656"
    },
    {
      "id": "5981",
      "nombre": "pepe",
      "numero": "666666666"
```

```
    }
  ]
}
```

Agenda-estilos

```
body {
  font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans
  color-scheme: light dark;
  color: rgba(255, 255, 255, 0.87);
  background-color: #242424;

}

.header{
  border: white 2px solid;
  width: 100%;
  height: 100px;
  justify-content: center;
  align-items: center;
  align-content: center;
  text-align: center;
}

.n-contactos{
  float: left;
  height: 100px;
  width: 50%;
  display: flex;
  align-items: center;
  text-align: center;
}
.usuario{
  float: right;
  height: 100px;
```

```
  width: 40%;
  display: flex;
  align-items: center;
  text-align: center;
}

.body{
  border: white 2px solid;
  width: 100%;
  height: 800px;
  display: flex;
  align-items: center;
  text-align: center;
  justify-content: center;
}
.boton{
  width: 200px;
  height: 100px;
}

ul{
  width: 100%;

}
li{
  margin: auto;
  background-color: rgb(94, 117, 138);
  border-radius: 15px;
  width: 80%;
  height: 60px;
  border: white solid 2px;
  align-content: center;
}

.caja-contacto *{
  margin: 0.2em;
```

```
  padding: 0.4em;
}

.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0, 0, 0, 0.5);
  display: flex;
  justify-content: center;
  align-items: center;
}

.modal-content {
  background: rgb(94, 87, 87);
  padding: 20px;
  border-radius: 8px;
  width: 400px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  position: relative;
}

.close-btn {
  background: red;
  color: white;
  border: none;
  padding: 5px 10px;
  border-radius: 5px;
  cursor: pointer;
  position: absolute;
  top: 10px;
  right: 10px;
}
```

```css
.toggle-usuario {
  width: 20%;
  float: right;
  background: white;
  color: #242424;
  border: 1px solid #ddd;
  border-radius: 8px;

}
```

Ejercicio carrito

App.jsx

```jsx
import { useState } from 'react'
import Menu from './componentes/Menu'
import Cuerpo from './componentes/Cuerpo'
import { Routes, Route } from 'react-router-dom';
import DetalleCarrito from './componentes/DetalleCarrito';
import DetalleProducto from './componentes/DetalleProducto';


function App() {

  const informacion = [
    { url: "/imagenes/manzana.jpg", nombre: "Manzana", precio: 5
    { url: "/imagenes/pera.jpg", nombre: "Pera", precio: 7 },
    { url: "/imagenes/platano.jpg", nombre: "Platano", precio: 4
  ];
  //const [total, setTotal] = useState(0);  Estado para el impor
  const [total, setTotal] = useState(0)
```

```
  const [productosJson, setProductosJson] = useState([]);

  return (
    <div className="App">

      <header className="App-header">
        {/* Pasar el total al menú superior */}
        <Menu
          total={total}
          setTotal={setTotal}
          productosJson={productosJson}
          setProductosJson={setProductosJson}
        />
      </header>
      <main>
        <Routes>
          <Route path='/'
            element={<Cuerpo total={total} setTotal={setTotal} 

          <Route path='/detalle-carrito' element={<DetalleCarri

          <Route path='/detalle-producto/:nombre' element={<Deta

        </Routes>

      </main>
    </div>
  );
}

export default App
```

Carrito-Header → toggle+eliminar

```jsx
import React, { useState } from "react";
import "../estilos/menu.css";
import { Link } from "react-router-dom";
import { obtenerCantidadTotal } from "../herramientas/buscarProd

// Componente MenuSuperior
const Menu = ({ total,setTotal, productosJson, setProductosJson
  const [carritoVisible, setCarritoVisible] = useState(false);

  const toggleCarrito = () => {
    setCarritoVisible(!carritoVisible);
  };

  // Función para eliminar un producto por su nombre (puedes car
  const eliminarProducto = (nombreProducto) => {
    // Filtrar el producto a eliminar
    const nuevosProductos = productosJson.filter((c) => c.nombre

    // Calcular el nuevo total sumando los precios de los produc
    const nuevoTotal = nuevosProductos.reduce((acc, producto) =>

    // Actualizar el estado con los nuevos productos y el nuevo
    setProductosJson(nuevosProductos);
    setTotal(nuevoTotal);
  };


  return (
    <div className="menu-superior">
      {/* Icono a la izquierda */}
      <img
        src="/imagenes/supermercado.png"
        alt="Supermercado"
        className="icono-supermercado"
      />
```

```
        <Link to="/"> Inicio</Link>
        <Link to="/detalle-carrito"> Detalle</Link>
        {/* Texto a la derecha */}
        <span className="carrito-texto">
          {obtenerCantidadTotal(productosJson)} : {total}€
        </span>

        {/* Botón para mostrar/ocultar carrito */}
        <button className="toggle-carrito" onClick={toggleCarrito
          🛒
        </button>

        {/* Carrito de productos */}
        {carritoVisible && (
          <div className="carrito-productos">
            <h4>Carrito</h4>
            {productosJson.length > 0 ? (
              <ul>
                {productosJson.map((producto, index) => (
                  <li key={index}>
                    {producto.nombre}:{producto.cantidad}{" "}
                    <button onClick={() => eliminarProducto(produc
                      Eliminar
                    </button>
                  </li>
                ))}
              </ul>
            ) : (
              <p>No hay productos en el carrito.</p>
            )}
          </div>
        )}
      </div>
    );
};
```

```
export default Menu;
```

Carrito-Añadir producto+mostrar opciones

Components/body

```
import '../estilos/cuerpo.css';
import { buscarProducto,incrementarCantidad } from '../herramie

// Datos de imágenes


// Componente Cuerpo
const Cuerpo = ({ total, setTotal, informacion, productosJson, 

  const AnadirProducto = (nombre, precio) => {

    setTotal(total + precio); // Actualiza el total

    if(buscarProducto(nombre,productosJson) === null){
      setProductosJson([...productosJson, {"nombre" : nombre, "
      console.log("Se añade un nuevo:", {"nombre" : nombre, "ca
    }else{

       setProductosJson(incrementarCantidad(productosJson,nombr

    }

  };

  return (
    <div className="container">
      {informacion.map((item, index) => (
        <div key={index}>
```

```
          <img src={item.url} alt="imagen" />
          <h3>{item.nombre}</h3>
          <p>Precio: {item.precio} Euros</p>
          <button onClick={() => AnadirProducto(item.nombre, ite
            Añadir al carrito
          </button>
        </div>
      ))}
    </div>
  );
};

export default Cuerpo;
```

Herramientas/buscarProducto.js

```
export function buscarProducto(nombre, informacion) {
  return (
    informacion.find(
      (producto) => producto.nombre.toLowerCase() === nombre.toL
    ) || null
  );
}

export function incrementarCantidad(informacion, nombre) {
  return informacion.map((producto) => {
    if (producto.nombre === nombre) {
      return { nombre: producto.nombre, cantidad: (producto.cant
    }

    return producto;
  });
}

export function obtenerCantidadTotal(informacion) {
  let total = 0;
```

```
  informacion.forEach((producto) => (total += producto.cantidad
  return total;
}
```

Carrito-CRUD →añadir, eliminar y editar cantidad

Components/DetalleCarrito

```
import { Link } from "react-router-dom";
import "../estilos/detalleCarrito.css";
import { buscarProducto } from "../herramientas/buscarProducto"

const DetalleCarrito = ({ productosJson, setProductosJson, infor
  // Función para añadir cantidad al producto
  const añadirCantidadProducto = (nombreProducto) => {
    setProductosJson((prevProductos) => {
      return prevProductos.map((producto) => {
        if (producto.nombre === nombreProducto) {
          // Incrementamos la cantidad del producto
          return { ...producto, cantidad: producto.cantidad + 1
        }
        return producto;
      });
    });
  };

  const eliminarCantidadProducto = (nombreProducto) => {
    setProductosJson((prevProductos) => {
      return prevProductos.map((producto) => {
        if (producto.cantidad>0 && producto.cantidad<10) {
          if (producto.nombre === nombreProducto) {
            // Incrementamos la cantidad del producto
            return { ...producto, cantidad: producto.cantidad -
          }
        }else{
```

```
            setProductosJson(prevProductos => {
              return prevProductos.filter(producto => producto.nor
            });
          }
          return producto;
        });
      });
    };

    // Función para modificar la cantidad del producto
    const modificarCantidadProducto = (nombreProducto) => {
      let cantidad = parseInt(prompt("Introduce cantidad: "));
      while (cantidad > 10) {
        cantidad = parseInt(prompt("Error: Introduce cantidad meno
      }
      setProductosJson((prevProductos) => {
        return prevProductos.map((producto) => {
          if (producto.nombre === nombreProducto) {
            // Incrementamos la cantidad del producto
            return { ...producto, cantidad: producto.cantidad + ca
          }
          return producto;
        });
      });
    };

    return (
      <div className="container-detalle-carrito">
        <ul>
          {productosJson.map((producto, index) => {
            let productoInformacion = buscarProducto(
              producto.nombre,
              informacion
            );

            return (
```

```
            <li key={index}>
              {productoInformacion.nombre} - {productoInformaci
              {producto.cantidad}
              <Link to={`/detalle-producto/${productoInformacio
                <img
                  src={productoInformacion.url}
                  alt={productoInformacion.nombre}
                />
              </Link>
              {/* Botón para añadir cantidad al producto */}
              <button
                onClick={() =>
                  añadirCantidadProducto(productoInformacion.no
                }
              >
                +
              </button>
              {/* Botón para eliminar el producto del carrito *
              <button
                onClick={() =>
                  eliminarCantidadProducto(productoInformacion.
                }
              >
                -
              </button>
              {/* Botón para modificar la cantidad del producto
              <button
                onClick={() =>
                  modificarCantidadProducto(
                    productoInformacion.nombre,
                    producto.cantidad + 1
                  )
                }
              >
                *
              </button>
```

```
            </li>
          );
        })}
      </ul>
    </div>
  );
};

export default DetalleCarrito;
```