crear en el front con react de los insumos para obterner informacion de ellos import { Formik, Form, Field, ErrorMes

```
import { useEffect, useState } from 'react'

import { useDispatch } from 'react-redux'
import * as Yup from 'yup'
import Select from 'react-select';
import { usePostSupplyMutation } from '../../context/Api/Common'
import {
  changeAction,
  closeModal,
  openModal,
  setAction,
  setWidth
} from '../../context/Slices/Modal/ModalSlice'
import Spinner from '../Spinner/Spinner'
import { toast } from 'react-toastify'

import clientAxios from '../../config/clientAxios'

const validationSchema = Yup.object().shape({
  name: Yup.string().required('Campo requerido'),
  dangerIndicators: Yup.string().required('Campo requerido'),
  useInstructions: Yup.string().required('Campo requerido'),
  advices: Yup.string().required('Campo requerido'),
  supplyType: Yup.number().required('Campo requerido'),
  sortingWord: Yup.number().required('Campo requerido'),
  quantity: Yup.number().required('Campo requerido'),
  averageCost: Yup.number().required('Campo requerido'),
  warehouseId: Yup.number().required('Campo requerido').moreThan(0, 'Debe elegir una bodega')
})

const getWarehause = () => {
  return new Promise((resolve, reject) => {
    clientAxios.get('/Warehause').then(
      (result) => {
        const warehouseId = result.data.map((Warehause) => ({
          label: Warehause.ubication,
          value: Warehause.id
        }))
        resolve(warehouseId)
      },
      (error) => {
        reject(error)
      }
    )
  })
}

//UnitMesure
const getUnitMesure = () => {
  return new Promise((resolve, reject) => {
    clientAxios.get('/UnitMesure').then(
      (result) => {
        const UnitMesure = result.data.map((unitMeasuresId) => ({
          'label': unitMeasuresId.name,
          'value': unitMeasuresId.id
        }));
        resolve(UnitMesure);
      },
      (error) => {
        reject(error);
      }
    );
  });
};

const getSupplyPictograms = () => {
  return new Promise((resolve, reject) => {
    clientAxios.get('/SupplyPictogrmas').then(
      (result) => {
        const SupplyPictogram = result.data.map((supplypictogram) => ({
          'label': supplypictogram.name,
          'value': supplypictogram.id
        }));
        resolve(SupplyPictogram);
      },
      (error) => {
        reject(error);
      }
    );
  });
};

const getSupplyCategories = () => {
  return new Promise((resolve, reject) => {
    clientAxios.get('/SupplyCategory').then(
      (result) => {
        const SupplyCategory = result.data.map((supplyCategory) => ({
          'label': supplyCategory.name,
          'value': supplyCategory.id
        }));
        resolve(SupplyCategory);
      },
      (error) => {
        reject(error);
      }
    );
  });
};



function CreateSupply() {
  const [unitMeasuresIdOptions, setunitmeaSuresIdOptions] = useState([]);
  const [SupplyPictogramsIdOptions, setSupplyPictogramsIdOptions] = useState([]);
  const [supplyCategoriesIdOptions, setsupplyCategoriesIdOptions] = useState([]);
  const [warehauseOptions, setWarehauseOptions] = useState([])

  // Add state variables for selected options
  const [selectedUnitMeasures, setSelectedUnitMeasures] = useState([]);
  const [selectedSupplyPictograms, setSelectedSupplyPictograms] = useState([]);
  const [selectedSupplyCategories, setSelectedSupplyCategories] = useState([]);



  const fetchOptions = () => {
    getUnitMesure().then((options) => {
      setunitmeaSuresIdOptions(options);
      setSelectedUnitMeasures([]); // Clear selected options
    });
    getSupplyPictograms().then((options) => {
      setSupplyPictogramsIdOptions(options);
      setSelectedSupplyPictograms([]); // Clear selected options
    });
    getSupplyCategories().then((options) => {
      setsupplyCategoriesIdOptions(options);
      setSelectedSupplyCategories([]); // Clear selected options
    });
    getWarehause().then((options) => {
      setWarehauseOptions(options)
    });
  };



  useEffect(() => {
    fetchOptions();
  }, []);



  const dispatch = useDispatch()
  const [createsupply, { error, isLoading }] = usePostSupplyMutation()

  const handleSubmit = async (values) => {
    if (isLoading) return <Spinner />

    // if (values.sortingWord === 1) {
    //   values.sortingWord = ' Peligro'
    // } else if (values.sortingWord === 2) {
    //   values.sortingWord = ' Atención'
    // }
    // if (values.supplyType === 1) {
    //   values.supplyType = ' Devolutivo'
    // } else if (values.supplyType === 2) {
    //   values.supplyType = ' Consumible'
    // }

    await createsupply(values)

    dispatch(changeAction())
    if (!error) {
      dispatch(closeModal())
    }
    toast.success('Insumo creado con exito', {
      autoClose: 1000

    })
  }




  return (

    <Formik

      initialValues={{
        name: '',
        dangerIndicators: '',
        useInstructions: '',
        advices: '',
        supplyType: 1,
        sortingWord: 1,
        quantity: 0,
        averageCost: 0,
        statedAt: true,
        unitMeasuresId: [],
        supplyPictogramsId: [],
        supplyCategoriesId: [],
        warehouseId: 0

      }}
      onSubmit={(values) => {
        console.log(values)
        handleSubmit(values)
      }}
      validationSchema={validationSchema}
    >
      {({ values, setFieldValue }) => (
      <Form>

        <div className="flex gap-5 grid-cols-5 mb-3">
          <div className="w-1/2">
            <label htmlFor="name" className="block mb-2 text-sm font-medium text-gray-900 dark:text-black">
             Nombre
            </label>
            <Field
              type="text"
              name="name"
              id="Nombre"
              className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus
              placeholder="Nombre"
            />
               <ErrorMessage
              name="name"

              component="div"
              className="text-red-500"
            />

          </div>
          <div className="w-1/4">
        <label htmlFor="supplyType" className="block mb-2 text-sm font-medium text-gray-900 dark:text-black">
        Tipo insumo
        </label>
        <Field
          as="select"
          name="supplyType"
          id="supplyType"
          className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus:b
        >
          <option value={1}>Devolutivo</option>
          <option value={2}>Consumible</option>
        </Field>
        <ErrorMessage
          name="supplyType"
          component="div"
          className="text-red-500"
        />
```

```
</div>
<div className="w-1/4">
  <label htmlFor="sortingWord" className="block mb-2 text-sm font-medium text-gray-900 dark:text-black">
    Tipo de peligrosidad
  </label>
  <Field
    as="select"
    name="sortingWord"
    id="sortingWord"
    className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus:b
  >
    <option value={1}>Peligro</option>
    <option value={2}>Atención</option>
  </Field>
  <ErrorMessage
    name="sortingWord"
    component="div"
    className="text-red-500"
  />
</div>
    <div className="w-1/7">
      <label htmlFor="quantity" className="block mb-2 text-sm font-medium text-gray-900 dark:text-black">
        Cantidad
      </label>
      <Field
        type="number"
        name="quantity"
        id="quantity"
        className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus
        placeholder="0"
      />
            <ErrorMessage
        name="quantity"

        component="div"
        className="text-red-500"
      />

    </div>
    <div className="w-1/7">
      <label htmlFor="averageCost" className="block mb-2 text-sm font-medium text-gray-900 dark:text-black">
        Costo promedio
      </label>
      <Field
        type="number"
        name="averageCost"
        id="averageCost"
        className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus
        placeholder="0"
      />
        <ErrorMessage
        name="averageCost"
        component="div"
        className="text-red-500"
      />

    </div>
    <div className='w-1/4 ml-2'>
    <label htmlFor="warehouseId">Bodega</label>
    <br />
```

```
          <Field name="warehouseId" as="select" className="bg-gray-50 border border-gray-300 text-gray-900 text-sm
l p-2.5">
            <option value={0}>Seleccione una bodega</option>
            {warehauseOptions.map(option => (
              <option key={option.value} value={option.value}>{option.label}</option>
            ))}
          </Field>
          <ErrorMessage
            name="warehouseId"
            component="div"
            className="text-red-500"
          />
        </div>
        </div>
        <div className="flex gap-5 grid-cols-5 mb-3">
      <div className="w-2/4">
        <label htmlFor="unitMeasuresId" className="block mb-2 text-sm font-medium text-gray-900 dark:text-black">
          Unidad de medida
        </label>
        <Field name="unitMeasuresId">
          {({ field }) => (
            <Select
              {...field}
              options={unitMeasuresIdOptions}
              isMulti
              value={selectedUnitMeasures}
              onChange={(selectedOptions) =>{
                setSelectedUnitMeasures(selectedOptions);
                setFieldValue(
                  "unitMeasuresId",
                  selectedOptions.map((option) => option.value)
                );
              }}
            />
          )}
        </Field>
      </div>

      <div className="w-2/4">
        <label htmlFor="supplyPictogramsId" className="block mb-2 text-sm font-medium text-gray-900 dark:text-blac
          Pictogramas de insumo
        </label>
        <Field name="supplyPictogramsId">
          {({ field }) => (
            <Select
              {...field}
              options={SupplyPictogramsIdOptions}
              isMulti
              value={selectedSupplyPictograms}
              onChange={(selectedOptions) => {
                setSelectedSupplyPictograms(selectedOptions);
                setFieldValue(
                  "supplyPictogramsId",
                  selectedOptions.map((option) => option.value)
                );
              }}
            />
          )}
        </Field>
      </div>
```

```
<div className="w-2/4">
  <label htmlFor="supplyCategoriesId" className="block mb-2 text-sm font-medium text-gray-900 dark:text-blac
    Categorías de insumo
  </label>
  <Field name="supplyCategoriesId">
    {({ field }) => (
      <Select
        {...field}
        options={supplyCategoriesIdOptions}
        isMulti
        value={selectedSupplyCategories}
        onChange={(selectedOptions) => {
          setSelectedSupplyCategories(selectedOptions);
          setFieldValue(
            "supplyCategoriesId",
            selectedOptions.map((option) => option.value)
          );
        }}
      />
    )}
    </Field>
</div>
</div>

  <div className="flex gap-5 grid-cols-5 mb-3">
    <div className="w-2/4">
      <label htmlFor="advices" className="block mb-2 text-sm font-medium text-gray-900 dark:text-black">
   Consejos
      </label>
      <Field
        as="textarea"
        type="text"
        name="advices"
        id="advices"
        className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus
        rows="6" cols="40"
        placeholder="Consejos"

      />
          <ErrorMessage
        name="advices"
        component="div"
        className="text-red-500"
      />

    </div>
    <div className="w-2/4">
      <label htmlFor="dangerIndicators" className="block mb-2 text-sm font-medium text-gray-900 dark:text-blac
      Indicadores de peligro
      </label>
      <Field
        as="textarea"
        type="text"
        name="dangerIndicators"
        id="dangerIndicators"
        className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus
        rows="6" cols="40"
        placeholder="Indicadores de peligro"
      />
```

```
            <ErrorMessage
          name="dangerIndicators"
          component="div"
          className="text-red-500"
        />


      </div>
      <div className="w-2/4">
        <label htmlFor="useInstructions" className="block mb-2 text-sm font-medium text-gray-900 dark:text-black"
         Instrucciones
        </label>
        <Field
          as="textarea"
          type="text"
          name="useInstructions"
          id="useInstructions"
          className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus
          rows="6" cols="40"
          placeholder="Instrucciones"
        />
              <ErrorMessage
          name="useInstructions"
          component="div"
          className="text-red-500"
        />

      </div>

      {/* <div className="w-1/8">
        <label htmlFor="campo2" className="block mb-2 text-sm font-medium text-gray-900 dark:text-black">
          Tipo insumo
        </label>
        <Field
          as="select"
          name="supplyType"
          id="supplyType"
          className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus
          placeholder="Tipo insumo"
        />
            <option value={1}>Devolutivo</option>
            <option value={2}>Consumible</option>

              <ErrorMessage
          name="supplyType"

          component="div"
          className="text-red-500"
        />

      </div> */}

     </div>

    <button
      type="submit"
      className="w-full text-white bg-custom-blue hover:bg-custom-blue-light focus:ring-4 focus:outline-none focus
5 py-2.5 text-center"
    >
      Crear Insumo
```

```
        </button>
        <button type="submit"></button>
      </Form>
      )}
    </Formik>
  )
}

export function CreateButtomSupply() {
  // ? Este bloque de codigo se usa para poder usar las funciones que estan declaradas en ModalSlice.js y se estan ex
  const dispatch = useDispatch()
  const handleOpen = () => {
    dispatch(setWidth({ width: '-[1500px]' }))
    dispatch(openModal({ title: 'Crear Insumo' }))
    dispatch(setAction({ action: 'creating' }))
  }
  // ?

  return (
    <button
      className="flex items-center justify-center border border-gray-400 text-white bg-custom-blue hover:bg-custom-
-lg text-sm px-4 py-2"
      type="button"
      onClick={() => handleOpen()}
    >
      <svg
        className="h-3.5 w-3.5 mr-2"
        fill="currentColor"
        viewBox="0 0 20 20"
        xmlns="http://www.w3.org/2000/svg"
        aria-hidden="true"
      >
        <path
          clipRule="evenodd"
          fillRule="evenodd"
          d="M10 3a1 1 0 011 1v5h5a1 1 0 110 2h-5v5a1 1 0 11-2 0v-5H4a1 1 0 110-2h5V4a1 1 0 011-1z"
        />
      </svg>
      Crear Insumo
    </button>
  )
}

export default CreateSupply
```

y el crear de la compra de insumos para que me hagas un formulario con los campos de la compra que son [id]
,[provider_id]
,[description]
,[entry_date]
,[security_file]
,[stated_at]
,[full_value]Nota: todo lo que tenga que ver con security_file no me lo vaya a quitar ni a cambiar.

donde el formulario va a tener la opcion de seleccionar los insumos que ya estan registrados(por eso comparto info
aiga la bodega asociada a ese insumo y el precio(que traiga el precio pero que en el campo lo pueda editar), pero co
ama supply_supplydetails donde esta la relacion de la compra y los insumos y la cantidad y el precio que le voy a da
necesito que me hagas esto en el codigo de CreateSupplyDetails, importeme de una ves lo que me tenga que impor
e pueda registrar la compra de insumos con los datos de la compra y los insumos con u respectiva cantidad y precio

```
import { Formik, Form, Field, ErrorMessage } from 'formik'
import { useDispatch } from 'react-redux'
import clientAxios from '../../config/clientAxios'
import * as Yup from 'yup'
import { usePostSupplyDetailsMutation } from '../../context/Api/Common'
import {
  changeAction,
  closeModal,
  openModal,
  setAction,
  setWidth
} from '../../context/Slices/Modal/ModalSlice'
import Spinner from '../Spinner/Spinner'
import { toast } from 'react-toastify'
import React, { useEffect, useMemo, useState } from 'react'
import { useTable, usePagination, useGlobalFilter } from 'react-table'
import { AiOutlineCloseCircle, AiOutlinePlusCircle } from 'react-icons/ai'

const validationSchema = Yup.object().shape({
  description: Yup.string(),
  supplyCost: Yup.number().required('Campo requerido'),
  quantity: Yup.string().required('Campo requerido'),
  // batch: Yup.string(),
  entryDate: Yup.date().required('Campo requerido'),
  expirationDate: Yup.date().required('Campo requerido'),
  supplyId: Yup.number().required('Campo requerido').moreThan(0, 'Debe elegir un insumo'),
  providerId: Yup.number().required('Campo requerido').moreThan(0, 'Debe elegir un proveedor'),
 security_file: Yup.string().required('Campo requerido'),
})

const getSupply = () => {
  return new Promise((resolve, reject) => {
    clientAxios.get('/Supply').then(
      (result) => {
        const supplyId = result.data.map((Supply) => ({
          label: Supply.name,
          value: Supply.id,
          price: Supply.averageCost
        }))
        resolve(supplyId)
      },
      (error) => {
        reject(error)
      }
    )
  })
}

const getProviders = async () => {
  return new Promise((resolve, reject) => {
    clientAxios.get('/Provider').then(
      (result) => {
        const providerId = result.data.map((Provider) => ({
          label: Provider.nameCompany,
          value: Provider.id
        }))
        resolve(providerId)
      },
      (error) => {
        reject(error)
      }
    )
  })
}



function CreateSupplyDetails () {
  const [dataApi, setDataApi] = useState([])
  const [listSupplies, setListSupplies] = useState([])
  const [listSuppliesWith, setlistSuppliesWith] = useState([])

  useEffect(() => {
    fetchOptions()
    get()
  }, [])

  const get = async () => {
    const { data } = await clientAxios('/supply')
    setDataApi(data)
  }

  const dispatch = useDispatch()
  const [createSupplyDetails, { isLoading }] = usePostSupplyDetailsMutation()
  const [previewImage, setPreviewImage] = useState(null)

  const [supplyOptions, setSupplyOptions] = useState([])
  const [providerOptions, setProviderOptions] = useState([])
  const [currentDate, setCurrentDate] = useState(new Date().toISOString().split('T')[0])
  const [dataSupplies, setDataSupplies] = useState(false)

  const [currentPage, setCurrentPage] = useState(1)
  const totalPages = 2

  const handleNextPage = () => {
    if (currentPage < totalPages) {
      setCurrentPage(currentPage + 1)
    }
  }

  const handlePrevPage = () => {
    if (currentPage > 1) {
      setCurrentPage(currentPage - 1)
    }
  }

  const fetchOptions = () => {
    getSupply().then((options) => {
      setSupplyOptions(options)
    })
    getProviders().then((options) => {
      setProviderOptions(options)
    })
  }

  useEffect(() => {
    fetchOptions()
    console.log(listSupplies)
  }, [])

  const handleSubmit = async (values) => {
    if (isLoading) return <Spinner />
    const formData = new FormData()
    formData.append('description', values.description)
    formData.append('supplyCost', values.supplyCost)
    formData.append('quantity', values.quantity)
    formData.append('entryDate', values.entryDate)
    formData.append('expirationDate', values.expirationDate)
    formData.append('suppliesId', values.suppliesId)
    formData.append('providerId', values.providerId)
    formData.append('security_file', values.security_file)
    for (const num of formData.entries()) {
      console.log("Datos del formulario a enviar al backend:", formData);
    }
    try {
      const response = await createSupplyDetails(formData)

      dispatch(changeAction())
      if (!response.error) {
        dispatch(closeModal())
      }

      toast.success('Compra de insumo creada con éxito', {
        autoClose: 1000
      })
    } catch (error) {
      console.error(error)
    }
  }
  const handleFileChange = event => {
    const file = event.target.files[0]
    if (file) {
      const reader = new FileReader()
      reader.onloadend = () => {
        setPreviewImage(reader.result)
      }
      reader.readAsDataURL(file)
    }
  }

  const addSupply = (cell) => {
    const quantityForSupply = quantities[cell.id] || 0
    const priceForSupply = prices[cell.id] || 0
    setlistSuppliesWith([...listSuppliesWith,
      {
        ...cell,
        quantityModified: parseInt(priceForSupply),
        priceModified: parseInt(quantityForSupply)
      }
    ])

    setTotal(total + parseInt(quantityForSupply))

    const existingSupply = listSupplies.find(supply => supply.id === cell.id)

    if (existingSupply) {
      const updatedSupplies = listSupplies.map(supply =>
        supply.id === cell.id ? { ...supply } : supply
      )
      setListSupplies(updatedSupplies)
    } else {
      setListSupplies([...listSupplies, { ...cell }])
    }
  }

  const deleteSupply = (cell) => {
    const updatedListSupplies = listSuppliesWith.filter(supply => supply.id !== cell)
    setlistSuppliesWith(updatedListSupplies)
  }

  const [quantities, setQuantities] = useState({})
  const [prices, setPrices] = useState({})
  const [total, setTotal] = useState(0)

  const handleQuantityChange = (supplyId, newQuantity) => {
    console.log(quantities)
    setQuantities({ ...quantities, [supplyId]: newQuantity })
  }

  const handlePriceChange = (priceId, newPrice) => {
    console.log(prices)
    setPrices({ ...prices, [priceId]: newPrice })
  }

  const columns = useMemo(() => [
    { Header: 'Nombre insumo', accessor: 'name' }
  ], [])

  const data = useMemo(() => (dataApi || []), [dataApi])

  const {
    getTableProps,
    getTableBodyProps,
    headerGroups,
    page,
    nextPage,
    previousPage,
    canNextPage,
    canPreviousPage,
    state,
    setGlobalFilter,
    prepareRow,
    rows: rowsTable
  } = useTable({
    data,
    columns
  },
  useGlobalFilter,
  usePagination)

  const { pageIndex, globalFilter } = state

  return (
    <Formik
      initialValues={{
        description: '',
        supplyCost: 0,
        quantity:"",
        entryDate: currentDate,
        expirationDate: '',
        suppliesId: [],
        providerId: 0,
        security_file:'',

      }}
      // validationSchema={validationSchema}

      onSubmit={values => {
        listSuppliesWith.map((supplyWith, index) => {
          console.log(values)
          console.log(supplyWith)
          console.log(supplyWith.id)
          console.log(index)
          const dataForm = {
            description: values.description,
            supplyCost: supplyWith.priceModified,
            quantity: supplyWith.quantityModified,
            entryDate: values.entryDate,
            expirationDate: values.expirationDate,
            supplyId: supplyWith.id,
            providerId: parseInt(values.providerId),
            warehouseId: parseInt(values.providerId)
          }

          return handleSubmit(dataForm)
        })

        // console.log({ ...values, typeProduct: typeProductSelect })
        // handleSubmit(dataForm)
      }}
    >
    {({ setFieldValue }) => (
      <Form>
        <div className="w-full">
          {currentPage === 1 && (
          <div className="">
          <div className="">
          <div className="flex gap-5 grid-cols-5 mb-3">
          <div className="w-4/4">
          <b>Codigo : 0001</b>
            </div>
            <div className="w-4/4">
              <b>Fecha: {new Date().toISOString().split('T')[0]}</b>
            </div>
          </div>
          <hr className="mb-4" />
          <div className="flex gap-5 grid-cols-2 mb-3">
          <div className="w-1/2 ml-2">
              <label htmlFor="entryDate">Fecha de entrada</label>
              <Field
                type="date"
                name="entryDate"
                id="entryDate"
                placeholder="Fecha de entrada"
                className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focu
                min={new Date().toISOString().split('T')[0]}
              />
        </div>

        <div className='w-1/2 mr-2'>
          <label htmlFor="expirationDate">Fecha de caducidad</label>
```

```
        <Field
          type="date"
          name="expirationDate"
          id="expirationDate"
          placeholder="Fecha de caducidad"
          className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus
          min={new Date().toISOString().split('T')[0]}
          />
      </div>
      </div>

        <div className="flex gap-5 grid-cols-2 mb-3">
        <div className='w-1/2 ml-2'>
        <label htmlFor="providerId">Proveedor</label>
        <br />
        <Field name="providerId" as="select" className="bg-gray-50 border border-gray-300 text-gray-900 text-sm ro
p-2.5">
          <option value={0}>Seleccione un proveedor</option>
          {providerOptions.map(option => (
            <option key={option.value} value={option.value}>{option.label}</option>
          ))}
        </Field>
        <ErrorMessage
          name="providerId"
          component="div"
          className="text-red-500"
        />
      </div>

      <div className='w-1/2 mr-2'>
      <label
          htmlFor="security_file"
          className="block mb-2 text-sm font-medium text-gray-900 dark:text-black"
        >
          Ficha de seguridad
        </label>
        {previewImage && <img src={previewImage} alt="Preview" width={100} height={100} />}
        <input
          type="file"
          name="security_file"
          id="security_file"
          placeholder="Ficha de seguridad"
          onChange={event => {
            setFieldValue('security_file', event.target.files[0])
            handleFileChange(event)
          }}
        />
        <ErrorMessage
    name="security_file"
    component="div"
    className="text-red-500"
  />
      </div>

      </div>
      <div className="flex gap-5 grid-cols-5 mb-3">
      <div className='w-full   mr-2'>
        <label htmlFor="description">Descripción</label>
        <Field
          as="textarea"
      type="text"
      name="description"
      id="description"
      placeholder="Descripción"
      className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-green-500 focus
      rows="6" cols="100"
  />
  <ErrorMessage
    name="description"
    component="div"
    className="text-red-500"
  />
</div>

  </div>

  <div className="flex gap-5 grid-cols-5 mb-2">
  <div className="flex flex-col items-end space-y-2 pt-6">
  <div className="flex w-full justify-between border-t border-gray-900/10 pt-2 md:w-1/2">
    <span className="font-bold">Total: ${total}</span>
    <span className="font-bold">
    </span>
  </div>
    </div>
</div>
  </div>
  </div>
  )}
</div>

{currentPage === 2 && (
<>
  <div className="w-full md:w-1/2">
    <div className="flex items-center">
      <label htmlFor="simple-search" className="sr-only">
        Buscar
      </label>
      <div className="relative w-full">
        <div className="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none">
          <svg
            aria-hidden="true"
            className="w-5 h-5 text-gray-500"
            fill="currentColor"
            viewBox="0 0 20 20"
            xmlns="http://www.w3.org/2000/svg"
          >
            <path
              fillRule="evenodd"
              d="M8 4a4 4 0 100 8 4 4 0 000-8zM2 8a6 6 0 1110.89 3.476l4.817 4.817a1 1 0 01-1.414 1.414l-4.816-4.81
              clipRule="evenodd"
            />
          </svg>
        </div>
        <input
          type="text"
          id="simple-search"
          className="bg-gray-50 border border-gray-400 text-gray-900 text-sm rounded-lg focus:ring-primary-500 f
          placeholder="Search"
          value={globalFilter || ''}
          onChange={e => setGlobalFilter(e.target.value)}
```

```
            />
          </div>
        </div>
      </div>
      <div className="overflow-x-auto rounded-xl border border-gray-400 mt-6">
        <table className="w-full text-sm text-left text-gray-500" {...getTableProps()}>
          <thead className="text-xs text-gray-700 uppercase bg-gray-50">
            {headerGroups.map(headerGroup => (
              <tr key={headerGroup.id} {...headerGroup.getHeaderGroupProps()}>
                {headerGroup.headers.map((column, index) => (
                  <th scope="col" className='px-6 py-3' key={`${column.id}-${index}`} {...column.getHeaderProps()}>
                    {column.render('Header')}
                  </th>
                ))}
                <th scope="col" className='px-6 py-3'>
                    Cantidad a modificar
                </th>
                <th scope="col" className='px-6 py-3'>
                    Precio a modificar
                </th>
                <th scope="col" className='px-6 py-3'>
                    Acciones
                </th>
              </tr>
            ))}
          </thead>
            {rowsTable.length === 0
              ? (
                <>
                  <p className='w-full text-center text-3xl font-bold ml-[170%] my-10'>No se encontraron registros con e
                </>
                )
              : (
                <>
                  {page.map(row => {
                    prepareRow(row)
                    return (
                      <tbody key={row.original.id} {...getTableBodyProps()}>
                        <tr
                          {...row.getRowProps()}
                          className="border-b border-gray-500"
                        >
                          {row.cells.map((cell, index) => {
                            return (
                              <React.Fragment key={`${cell.row.original.id}-${index}`}>
                                <td {...cell.getCellProps()} className="px-4 py-3">
                                  {typeof cell.value === 'function' ? cell.value(cell) : cell.render('Cell')}
                                </td>
                                {index === 0 &&
                                  <>
                                    <td className="px-6 py-4 place-content-center" key={`${cell.row.original.id}-price`}>
                                      <Field
                                        type="text"
                                        className="w-full px-4 py-2 leading-tight text-gray-700 border rounded-lg focus:outline-nor
                                        value={prices[cell.row.original.id]}
                                        onChange={(e) => handlePriceChange(cell.row.original.id, e.target.value)}
                                      />
                                    </td>
                                    <td className="px-6 py-4 place-content-center" key={`${cell.row.original.id}-quantity`}>
                                      <Field
                              type="text"
                              className="w-full px-4 py-2 leading-tight text-gray-700 border rounded-lg focus:outline-nor
                              value={quantities[cell.row.original.id]}
                              onChange={(e) => handleQuantityChange(cell.row.original.id, e.target.value)}
                            />
                          </td>
                          <td className="px-6 py-4 grid grid-cols-3  place-content-center" key={`${cell.row.original.id}-a
                            <button type="button" onClick={() => addSupply(cell.row.original)}>
                              <AiOutlinePlusCircle alt="Icono agregar " title="Agregar " className="h-6 w-6 mr-2" />
                            </button>
                            <button type="button" onClick={() => deleteSupply(cell.row.original.id)}>
                              <AiOutlineCloseCircle alt="Icono eliminar " title="Eliminar " className="h-6 w-6 mr-2" />
                            </button>
                          </td>
                        </>
                      }
                    </React.Fragment>)
                  })}
                </tr>
              </tbody>
            )
          })}
        </>
        )
      }
    </table>
    <nav
      className="flex flex-col md:flex-row justify-between items-start md:items-center space-y-3 md:space-y-0 p-4
    >
      <span className="text-sm font-normal text-gray-500">
        Página {' '}
        <span className="font-semibold text-gray-900">{pageIndex + 1}</span>
      </span>
      <ul className="inline-flex items-stretch -space-x-px">
        <li>
          <button
            className="flex items-center justify-center h-full py-1.5 px-3 ml-0 text-gray-500 bg-white rounded-l-lg bo
700"

            onClick={() => previousPage()}
            disabled={!canPreviousPage}
          >
            <span className="sr-only">Anterior</span>
            <svg
              className="w-5 h-5"
              aria-hidden="true"
              fill="currentColor"
              viewBox="0 0 20 20"
              xmlns="http://www.w3.org/2000/svg"
            >
              <path
                fillRule="evenodd"
                d="M12.707 5.293a1 1 0 010 1.414L9.414 10l3.293 3.293a1 1 0 01-1.414 1.414l-4-4a1 1 0 010-1.414l4-4a
                clipRule="evenodd"
              />
            </svg>
          </button>
        </li>
        <li>
          <a
            className="flex items-center justify-center text-sm py-2 px-3 leading-tight text-gray-500 bg-white border
```

```
              >
                -
              </a>
            </li>
            <li>
              <button
                className="flex items-center justify-center h-full py-1.5 px-3 leading-tight text-gray-500 bg-white rounde
ext-gray-700"
                onClick={() => nextPage()}
                disabled={!canNextPage}
              >
                <span className="sr-only">Siguiente</span>
                <svg
                  className="w-5 h-5"
                  aria-hidden="true"
                  fill="currentColor"
                  viewBox="0 0 20 20"
                  xmlns="http://www.w3.org/2000/svg"
                >
                  <path
                    fillRule="evenodd"
                    d="M7.293 14.707a1 1 0 010-1.414L10.586 10 7.293 6.707a1 1 0 011.414-1.414l4 4a1 1 0 010 1.414l-4 4
                    clipRule="evenodd"
                  />
                </svg>
              </button>
            </li>
          </ul>
        </nav>
      </div>

    <p className='text-xl font-semibold text-gray-900 lg:text-2xl mt-8'>Lista de insumos de la compra</p>

    <div className="overflow-x-auto rounded-xl border border-gray-400 mt-6">
      <table className="w-full text-sm text-left text-gray-500" {...getTableProps()}>
        <thead className="text-xs text-gray-700 uppercase bg-gray-50">
          {headerGroups.map(headerGroup => (
            <tr key={headerGroup.id} {...headerGroup.getHeaderGroupProps()}>
              {headerGroup.headers.map((column, index) => (
                <th scope="col" className='px-3 py-3' key={`${column.id}-${index}`} {...column.getHeaderProps()}>
                  {column.render('Header')}
                </th>
              ))}
              <th scope="col" className='px-6 py-3'>
                  Cantidad a modificar
                </th>
                <th scope="col" className='px-6 py-3'>
                  Precio a modificar
                </th>
            </tr>
          ))}
        </thead>
        <>
        <tbody>
          {listSuppliesWith.map(listSupply => {
            return (
              <tr
                key={listSupply.id}
                className="border-b border-gray-500"
              >
```

```
                {console.log(listSupply)}
                <td className="px-4 py-3">{listSupply.name}</td>
                <td className="px-4 py-3">{listSupply.quantityModified}</td>
                <td className="px-4 py-3">{listSupply.priceModified}</td>
              </tr>
            )
          })}
        </tbody>
        </>
      </table>
    </div>
    </>
      )}
      <div className='space-x-5 mt-8'>
      <button type='button' onClick={handlePrevPage} disabled={currentPage === 1} className={`${currentPage ===
 focus:ring-4 focus:outline-none focus:bg-custom-blue-light font-medium rounded-lg text-sm px-5 py-2.5 text-center
        Página anterior
      </button>
      <button type='button' onClick={handleNextPage} disabled={currentPage === totalPages} className={`${curren
-custom-blue-light focus:ring-4 focus:outline-none focus:bg-custom-blue-light font-medium rounded-lg text-sm px-5
        Siguiente página
      </button>
    </div>

    <button
      type="submit"
      className="w-full text-white bg-custom-blue hover:bg-custom-blue-light focus:ring-4 focus:outline-none focus:
text-center mt-6"
    >
      Crear compra de insumos
    </button>
    </Form>
  )}
  </Formik>
 )
}

export function CreateButtomSupplyDetails () {
 // ? Este bloque de codigo se usa para poder usar las funciones que estan declaradas en ModalSlice.js y se estan ex
 const dispatch = useDispatch()
 const handleOpen = () => {
  dispatch(setWidth({ width: '1500px' }))
  dispatch(openModal({ title: 'Crear compra de insumo' }))
  dispatch(setAction({ action: 'creating' }))
 }
 // ?

 return (
  <button
   className="flex items-center justify-center border border-gray-400 text-white bg-custom-blue hover:bg-custom-
-lg text-sm px-4 py-2"
    type="button"
    onClick={() => handleOpen()}
  >
    <svg
     className="h-3.5 w-3.5 mr-2"
     fill="currentColor"
     viewBox="0 0 20 20"
     xmlns="http://www.w3.org/2000/svg"
     aria-hidden="true"
```

```
      >
        <path
          clipRule="evenodd"
          fillRule="evenodd"
          d="M10 3a1 1 0 011 1v5h5a1 1 0 110 2h-5v5a1 1 0 11-2 0v-5H4a1 1 0 110-2h5V4a1 1 0 011-1z"
        />
      </svg>
      Crear compra de insumo
    </button>
  )
}

export default CreateSupplyDetails
```