Validando Gastos

- **Ahora** es realizar una lógica para que cuando se le dé al submit de añadir gasto, se validen los campos y en caso de no ser válido mostrar un mensaje
- Pongo el onSubmit en el form y le introduzco la funcion que crearé antes del return del componente, handleSubmit
- Le coloco el preventDefault para evitar el refresh y un console.log para comprobar que cuando doy clic al añadir gasto envía diciendo 'Añadiendo formulario'

```
  const handleSubmit = (e) =>{
    e.preventDefault()
    console.log('Añadiendo Formulario')
  }

return (
  <div className="modal">
      <div className="cerrar-modal">
          <img src={CerrarBtn} alt="cerrar modal"
          onClick={ocultarModal}/>
      </div>
      <form onSubmit={handleSubmit}
      className={`formulario ${animarModal ? "animar":'cerrar'}`}>
```

- Para la validación coloco las variables del state en un arreglo y uso .includes, para decir que si alguno de los tres campos está vacío ejecute una accíon.
- Uso el console.log para comprobar que al momento de validar cumple si alguno de los campos no estan rellenados
- Le añado un return para que no ejecute las siguientes lineas

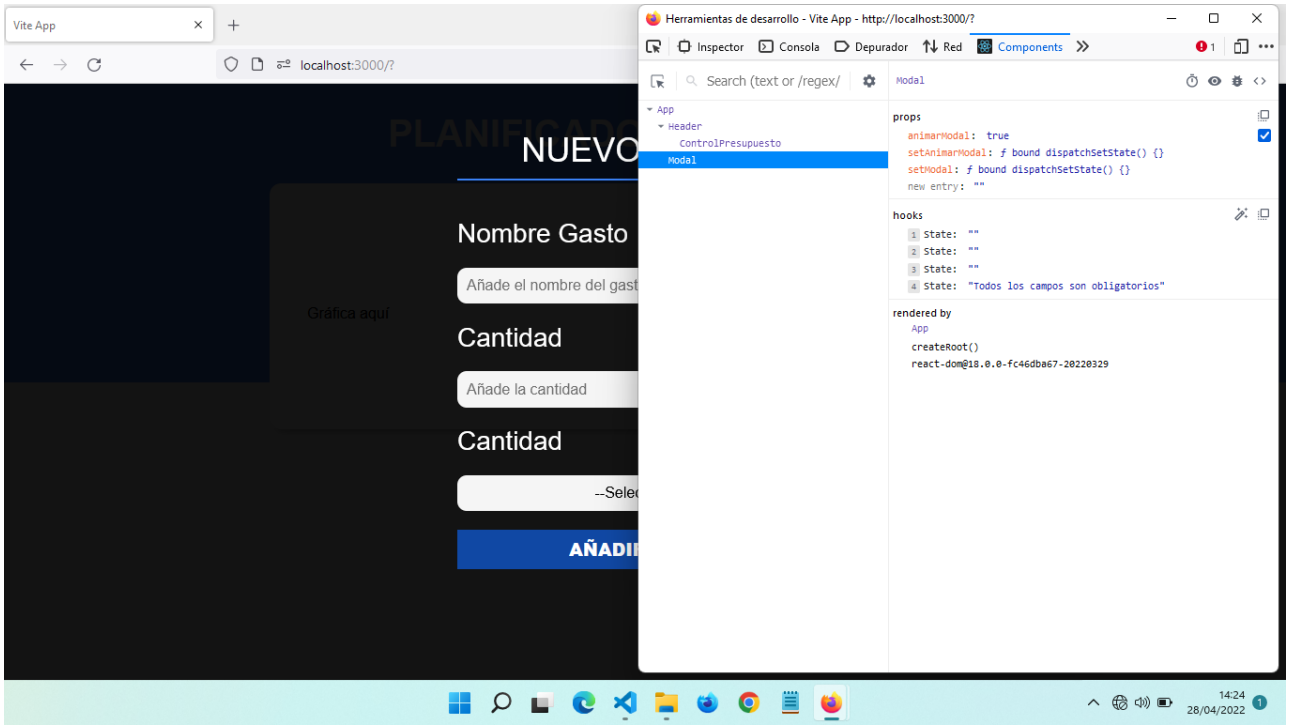
```
const handleSubmit = (e) =>{
     e.preventDefault()
     if([nombre, cantidad, categoria].includes('')){
       console.log('Falló la validación')
       return
     }
   }
```

- Ahora en lugar de ese console.log, voy a crear un nuevo state más, mensaje y setMensaje

```
  const handleSubmit = (e) =>{
     e.preventDefault()
     if([nombre, cantidad, categoria].includes('')){
```

```
            setMensaje('Todos los campos son obligatorios')
        }
    }
```

- Puedo comprobar el state en components de devtools dando clic a "Añadir Gasto"



- Ahora puedo reutilizar el componente Mensaje.
    - Tiene un children, que es todo lo que introduzca en el componente
    - Tiene un tipo, que tambien va a ser error, siendo la clase de CSS

```
import React from 'react'

const Mensaje = ({children, tipo}) => {
  return (
    <div className= {`alerta ${tipo}`}>
        {children}
    </div>
  )
}

export default Mensaje
```

- Lo importo.
- Coloco el mensaje después del legend de gasto , y le voy a decir que
    - cuando mensaje esté en true muestre el componente de mensaje
    - le paso por props el tipo error y le coloco el mensaje en el children

- Le paso el mensaje que estoy seteando con setMensaje

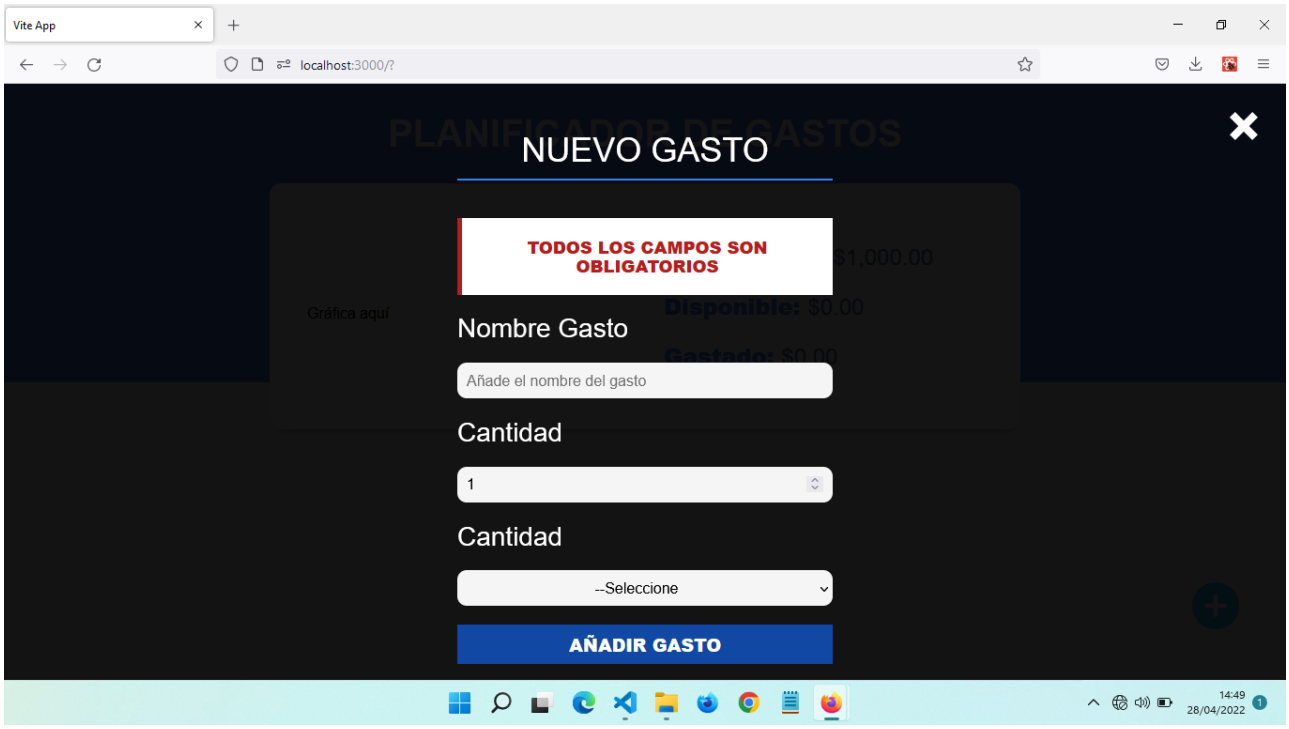
```
const handleSubmit = (e) =>{
  e.preventDefault()
  if([nombre, cantidad, categoria].includes('')){
    setMensaje('Todos los campos son obligatorios')
  }
}

return (
  <div className="modal">
      <div className="cerrar-modal">
          <img src={CerrarBtn} alt="cerrar modal"
          onClick={ocultarModal}/>
      </div>
      <form onSubmit={handleSubmit}
      className={`formulario ${animarModal ? "animar":'cerrar'}`}>
        <legend>Nuevo Gasto</legend>

        {mensaje && <Mensaje tipo="error">{mensaje}</Mensaje>}

        <div className="campo">
          <label htmlFor="nombre">Nombre Gasto</label>
```

- El resultado es este



- Para quitar este mensaje al cabo de unos segundos puedo colocar un setTimeOut y setear el mensaje a string vacío con setMensaje

- Ahora en App.jsx creo una función antes del return que se llame guardarGasto y se la paso por props al modal, con el console.log de gasto

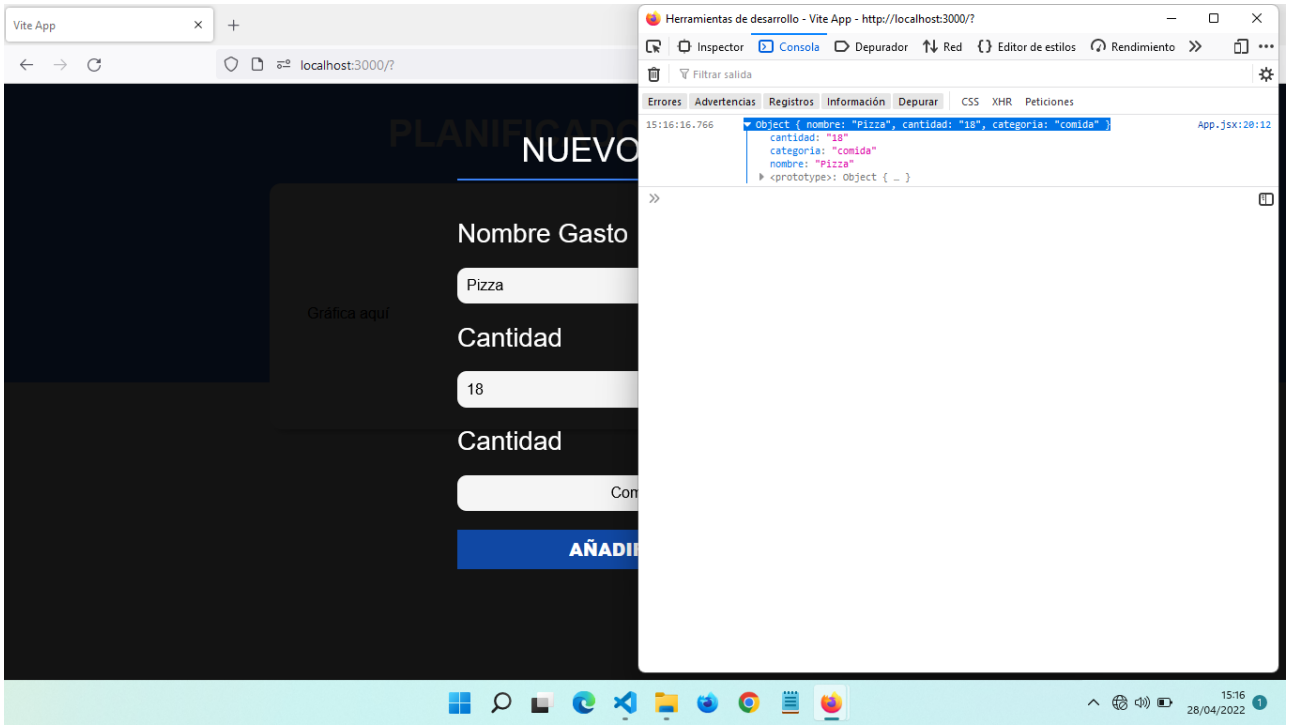
```
const guardarGasto = gasto =>{
    console.log(gasto)
}
```

- Se lo agrego al handleSubmit fuera de la validación, pongo un return y genero el objeto nombre, cantidad y categoria. Lo imprime en consola

```
const handleSubmit = (e) =>{
  e.preventDefault()
  if([nombre, cantidad, categoria].includes('')){
    setMensaje('Todos los campos son obligatorios')

    setTimeout(()=>{
      setMensaje('')
    },2000)
    return
  }
  guardarGasto({nombre, cantidad,categoria})
}
```

- Lo puedo ver en consola



- Ya tengo el objeto
- Ahora puedo crear otro state con gastos, setGastos con valor inicial de un arreglo vacío
- Entonces puedo usar setGastos para agregar los gastos en App.jsx
- Le paso del hijo al padre, pero para hacer iteraciones necesito un id único
- Veamos cómo hacerlo