Manejo de formulario para los ítems

En esta sección, se procederá a crear un formulario en la aplicación web para ingresar nuevos elementos a la factura.

Pasos 57 al 61

Sigue estos pasos:

Paso 57: La estructura del formulario

El formulario se añade dentro del componente **InvoiceApp.jsx**, después de mostrar el total de la factura. Para aquello:

* Se utiliza la etiqueta **<form>** para encapsular los campos del formulario.
* Se añaden campos de entrada **<input>** para ingresar el nombre del producto, el precio y la cantidad.
* Se utiliza el atributo **type** para especificar que los campos sean de tipo **"text"** y el atributo **name** en los campos de entrada para nombrar el campo, el cual va a coincidir con los nombres de los atributos de los elementos de la lista de ítems en **invoice.js** (**product**, **price** y **quantity**).
* Se utiliza el atributo **placeholder** para proporcionar texto de ayuda dentro de los campos de entrada.

**Nota:** En React, la etiqueta de cierre para elementos como **<input>**, **<img>**, **<br>**, **<hr>**, y otros elementos que no tienen contenido, se hace mediante la auto-cierre de la etiqueta, es decir, agregando un slash (**/**) justo antes del cierre de la etiqueta de apertura. Por ejemplo: **<input />**.

Todo esto queda de la siguiente manera:

// import { invoice } from "../data/invoice"
import { getInvoice } from "./services/getInvoice"
import { ClientView } from "./components/ClientView";
import { CompanyView } from "./components/CompanyView";
import { InvoiceView } from "./components/InvoiceView";
import { ListItemsView } from "./components/ListItemsView";
import { TotalView } from "./components/TotalView";

export const InvoiceApp = () => {

    const { total, id, name, client, company, items } = getInvoice()

    return (
        <>
            <div className="container">
                <div className="card my-3">
                    <div className="card-header">
                        Ejemplo Factura
                    </div>
                    <div className="card-body">
                        <InvoiceView id={id} name={name} />

                        <div className="row my-3">
                            <div className="col">
                                <ClientView title="Datos del cliente" client={client} />
                            </div>

                            <div className="col">
                                <CompanyView title="Datos de la empresa" company={company} />
                            </div>
                        </div>

                        <ListItemsView title="Productos de la factura" items={items} />

                        <TotalView total={total} />

                        <form>
                            <input
                                type="text"
                                name="product"
                                placeholder="Producto" />
                            <input
                                type="text"
                                name="price"
                                placeholder="Precio" />
                            <input
                                type="text"
                                name="quantity"
                                placeholder="Cantidad" />
                        </form>
                    </div>
                </div>
            </div>
        </>
    )
}



// import { invoice } from "../data/invoice"
import { getInvoice } from "./services/getInvoice"
import { ClientView } from "./components/ClientView";
import { CompanyView } from "./components/CompanyView";
import { InvoiceView } from "./components/InvoiceView";
import { ListItemsView } from "./components/ListItemsView";
import { TotalView } from "./components/TotalView";

export const InvoiceApp = () => {

    const { total, id, name, client, company, items } = getInvoice()

    return (
        <>
            <div className="container">
                <div className="card my-3">
                    <div className="card-header">
                        Ejemplo Factura
                    </div>
                    <div className="card-body">
                        <InvoiceView id={id} name={name} />

                        <div className="row my-3">
                            <div className="col">
                                <ClientView title="Datos del cliente" client={client} />
                            </div>

                            <div className="col">
                                <CompanyView title="Datos de la empresa" company={company} />
                            </div>
                        </div>

                        <ListItemsView title="Productos de la factura" items={items} />

                        <TotalView total={total} />

                        <form>
                            <input
                                type="text"
                                name="product"
                                placeholder="Producto" />
                            <input
                                type="text"
                                name="price"
                                placeholder="Precio" />
                            <input
                                type="text"
                                name="quantity"
                                placeholder="Cantidad" />
                        </form>
                    </div>
                </div>
            </div>
        </>
    )
}



Paso 58: Aplicar estilos a los elementos del formulario

Al agregar la clase **form-control** a cada uno de los **<input>** (con el atributo **className**), se aplican los estilos de Bootstrap para que los campos de entrada tengan un aspecto más uniforme y estilizado.

Además, al agregar la clase **my-3**, se aplica un margen tanto arriba como abajo de cada campo de entrada para evitar que queden demasiado pegados entre sí o con otros elementos cercanos.

**Nota:** Recuerda que puedes activar el multicursor en Visual Studio Code manteniendo pulsada la tecla **ATL** y luego haz clic en las zonas donde deseas agregar el mismo código que vas a escribir.

// import { invoice } from "../data/invoice"
import { getInvoice } from "./services/getInvoice"
import { ClientView } from "./components/ClientView";
import { CompanyView } from "./components/CompanyView";
import { InvoiceView } from "./components/InvoiceView";
import { ListItemsView } from "./components/ListItemsView";
import { TotalView } from "./components/TotalView";

export const InvoiceApp = () => {

    const { total, id, name, client, company, items } = getInvoice()

    return (
        <>
            <div className="container">
                <div className="card my-3">
                    <div className="card-header">
                        Ejemplo Factura
                    </div>
                    <div className="card-body">
                        <InvoiceView id={id} name={name} />

                        <div className="row my-3">
                            <div className="col">
                                <ClientView title="Datos del cliente" client={client} />
                            </div>

                            <div className="col">
                                <CompanyView title="Datos de la empresa" company={company} />
                            </div>
                        </div>

                        <ListItemsView title="Productos de la factura" items={items} />

                        <TotalView total={total} />

                        <form>
                            <input
                                type="text"
                                name="product"
                                placeholder="Producto" className="form-control my-3" />
                            <input
                                type="text"
                                name="price"
                                placeholder="Precio" className="form-control my-3" />
                            <input
                                type="text"
                                name="quantity"
                                placeholder="Cantidad" className="form-control my-3" />
                        </form>
                    </div>
                </div>
            </div>
        </>
    )
}



// import { invoice } from "../data/invoice"
import { getInvoice } from "./services/getInvoice"
import { ClientView } from "./components/ClientView";
import { CompanyView } from "./components/CompanyView";
import { InvoiceView } from "./components/InvoiceView";
import { ListItemsView } from "./components/ListItemsView";
import { TotalView } from "./components/TotalView";

export const InvoiceApp = () => {

    const { total, id, name, client, company, items } = getInvoice()

    return (
        <>
            <div className="container">
                <div className="card my-3">
                    <div className="card-header">
                        Ejemplo Factura
                    </div>
                    <div className="card-body">
                        <InvoiceView id={id} name={name} />

                        <div className="row my-3">
                            <div className="col">
                                <ClientView title="Datos del cliente" client={client} />
                            </div>

                            <div className="col">
                                <CompanyView title="Datos de la empresa" company={company} />
                            </div>
                        </div>

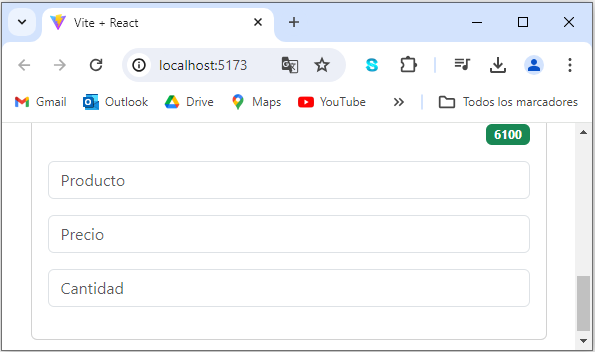
                        <ListItemsView title="Productos de la factura" items={items} />

                        <TotalView total={total} />

                        <form>
                            <input
                                type="text"
                                name="product"
                                placeholder="Producto" className="form-control my-3" />
                            <input
                                type="text"
                                name="price"
                                placeholder="Precio" className="form-control my-3" />
                            <input
                                type="text"
                                name="quantity"
                                placeholder="Cantidad" className="form-control my-3" />
                        </form>
                    </div>
                </div>
            </div>
        </>
    )
}



En resumen, al agregar estas clases a los campos de entrada del formulario, se mejora su apariencia y su disposición dentro de la página, lo que contribuye a una mejor experiencia de usuario.

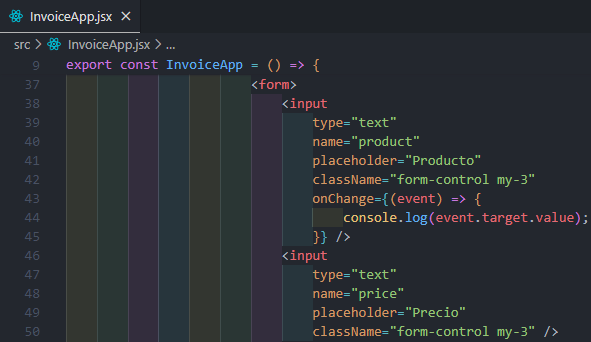


Paso 59: Detectar cambios en los campos del formulario

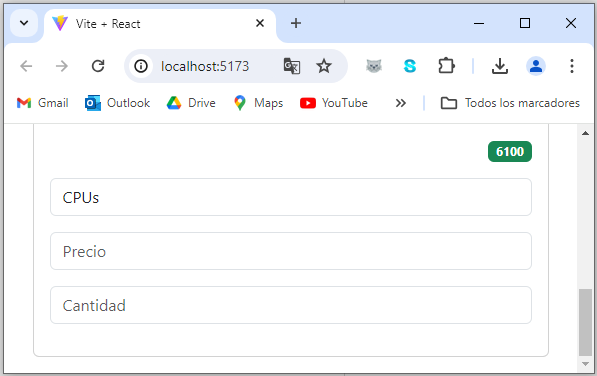
Implementar el evento **onChange** en los campos de entrada del formulario va a permitir detectar y manejar los cambios en tiempo real mientras el usuario escribe en los campos. En React, cada campo de entrada puede manejar eventos, como **onChange**, que se activa cada vez que el valor del campo cambia.

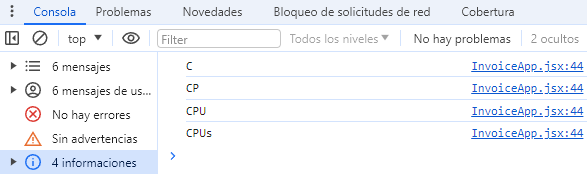
Proceso de implementación:

1. Se agrega el evento **onChange** al campo de entrada **product** del formulario. Este evento se activa cuando el valor del campo cambia.
2. Al producirse un cambio en el campo, se ejecuta una función. Esta función recibe un objeto evento (**event**) que contiene el valor actual del campo.
3. Para obtener el valor actual del campo, se utiliza **event.target.value**. Luego se utiliza **console.log** para imprimir el valor actual del campo en la consola del navegador.



Al escribir en el campo producto, cada vez que se modifique su valor, se imprimirá el valor actual en la consola del navegador. Esto permite que se pueda ver en tiempo real lo que el usuario está escribiendo y realizar acciones basadas en esos cambios.



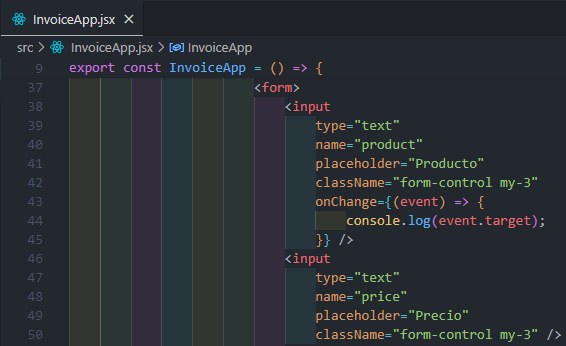


Paso 60: Utilizar las propiedades value y target

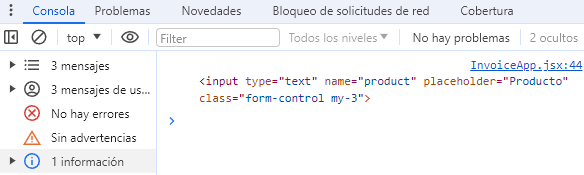
Es importante comprender la diferencia entre las propiedades **value** y **target** al trabajar con eventos de cambio (**onChange**) en los campos de entrada.

* **Value:** Se refiere al valor actual del campo de entrada. En el contexto del evento **onChange**, **event.target.value** proporciona acceso directo al valor actual que el usuario ha ingresado o modificado en el campo.
* **Target:** Hace referencia al elemento que activó el evento. En este caso, cuando se produce un cambio en el campo de entrada, **event.target** representa el elemento **<input>** en sí mismo, con todos sus atributos, como **type**, **name**, **placeholder**, entre otros.

Implementa la propiedad **target** cuando se realiza algún cambio en el campo del formulario:

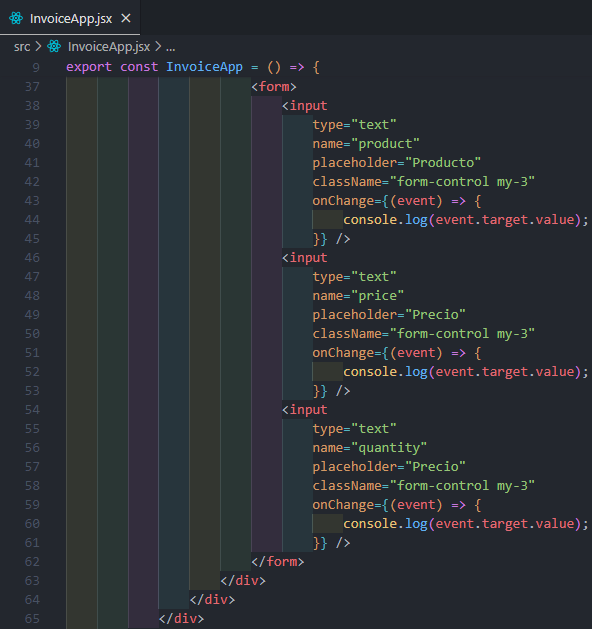


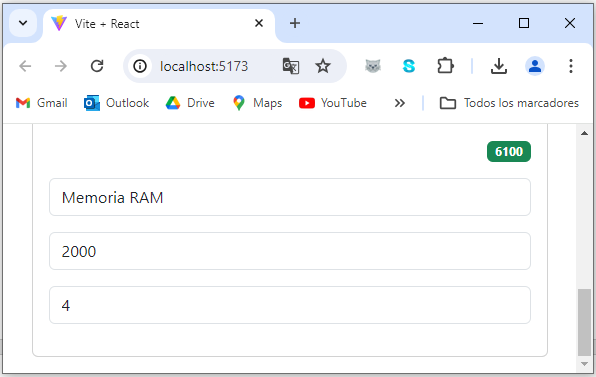
En el navegador, al momento de escribir un carácter (cualquier letra, número o símbolo) en el campo **product**, al observar la consola del navegador, se puede ver el objeto completo del elemento **<input>**, lo que facilita la comprensión de su estructura y los datos asociados a él.

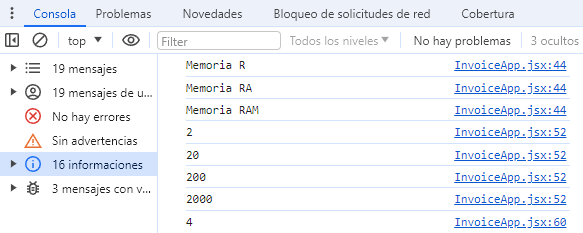


Paso 61: Aplicar el evento onChange en los campos restantes

Extiende la funcionalidad de imprimir en consola a los otros campos del formulario. Al introducir valores en cualquier campo de texto desde el navegador, estos se imprimirán en la consola del navegador.







Recordar que en React, la filosofía fundamental es que está basado en el estado de la página, lo que implica que la información de la página se debe almacenar en algún lugar para que pueda ser accedida y manipulada fácilmente.