



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE MILPA ALTA II

ASIGNATURA:

MEAN STACK FOR FRONT – END

TEMA:

Crud de registros con ngModel y ts

ELABORA:

POZOS RIVERA ALEJANDRO

NÚMERO DE CONTROL:

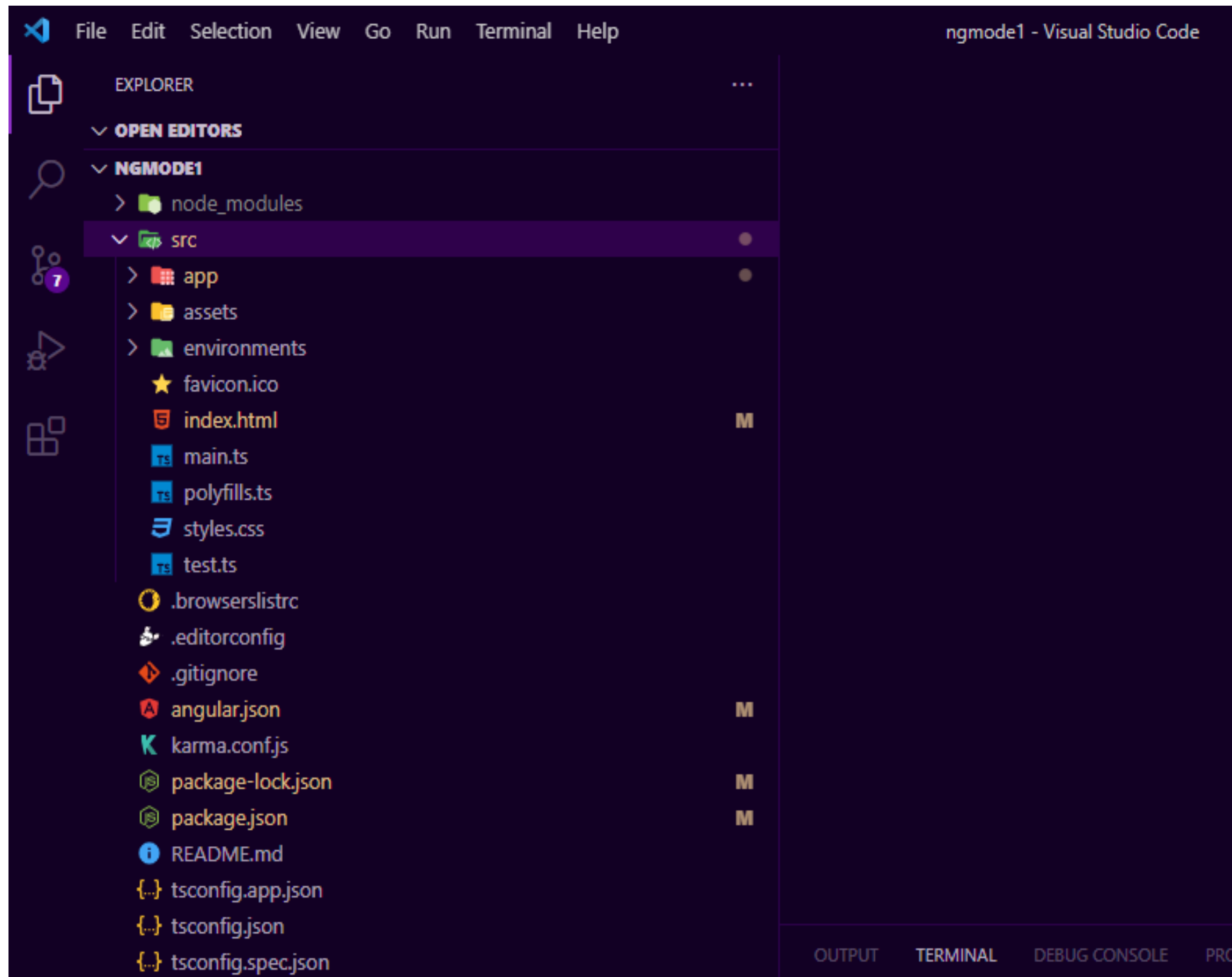
171190025

PROFESOR:

ROLDAN AQUINO SEGURA

09 DE JUNIO DE 2021

1.- En primer lugar, ya debemos de tener el proyecto de angular construido.



2.- Ahora vamos a ir al apartado de src → app → app.component.html, para empezar a construir nuestra interfaz de usuario. Para el apartado del encabezado lo vamos a albergar dentro de un div con la clase container, un h1 con display-4 y con un pequeño estilo para poder centrar el texto.

```
app.component.html M X
src > app > app.component.html > div.container > div.row > table.table.table-striped.table-secondary.lead > thead
1  <div class="container mt-3">
2    <div class="row mt-3">
3      <div class="col mt-3" style="text-align:center;">
4        <h1 class="display-4">
5          {{title}}
6        </h1>
7        <h3>Este crud simula el agregado de inventario de ropa</h3>
8      </div>
9    </div>
10 </div>
```

3.- El siguiente apartado es donde contendrá nuestra tabla con los datos que vaya a albergar.

```
<div class="container">
  <div class="row">
    <table class="table table-striped table-secondary lead" border="2" style="text-align: center;">
      <thead>
        <tr>
          <th>Id</th>
          <th>Nombre</th>
          <th>Precio</th>
          <th>Borrar</th>
          <th>Seleccionar</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let articulos of inventario">
          <td>{{articulos.id}}</td>
          <td>{{articulos.nombre}}</td>
          <td>{{articulos.precio}}</td>
          <td>
            <button class="btn btn-outline-danger" (click)="eliminar(articulos.id)"><i
              class="fas fa-trash-alt"></i>Borrar</button>
          </td>
          <td>
            <button class="btn btn-outline-primary" (click)="seleccionar(articulos)">Seleccionar</button>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

4.- En el apartado de la tabla existen 2 botones los cuales son “Borrar” y “Seleccionar”.

```
<button class="btn btn-outline-danger" (click)="eliminar(articulos.id)"><i  
    class="fas fa-trash-alt"></i>Borrar</button>  
</td>  
<td>  
    <button class="btn btn-outline-primary" (click)="seleccionar(articulos)">Seleccionar</button>
```

Los cuales para que en primera instancia realicen sus funciones dentro de ellos está declarado el método “click” que es el que nos va a ayudar a captar cuando el usuario presione el botón. El botón “Borrar” va a recibir la función ‘eliminar(articulos.id)’, como se muestra. Con el método splice nos ayudara a eliminar el elemento que marque el index dentro de nuestra iteración.

```
public eliminar(id: number) {  
    for (let index = 0; index < this.inventario.length; index++) {  
        if (this.inventario[index].id === id) {  
            this.inventario.splice(index, 1);  
            alert("Eliminado con exito");  
            break;  
        }  
    }  
}
```

El botón “Seleccionar” va a recibir todos los artículos que contenga nuestro array artículos. Los cuales se asignaran al array llamado formulario.

```
public seleccionar(articulos: any) {  
    this.formulario.id = articulos.id;  
    this.formulario.nombre = articulos.nombre;  
    this.formulario.precio = articulos.precio;  
}
```

```
public formulario: any = {  
  id: null,  
  nombre: null,  
  precio: null  
};
```

Los primeros objetos de nuestro array inventario son pantalón y chamarra. Cabe mencionar que cada objeto contara con tres atributos lo cuales son id, nombre y precio.

```
public inventario: any = [  
  { id: 1, nombre: "Pantalon", precio: 250 },  
  { id: 2, nombre: "Chamarra", precio: 500 }  
];
```

5.- En la parte de nuestro formulario contaremos con un div que tendrá la clase container para que albergue el contenido de nuestro formulario, por otro lado, tendremos un label donde tendrá el nombre de lo que vamos a pedir y un input donde colocara la información que se pida según el caso, los cuales serán tres uno para id, nombre y precio. Y en la ultima parte tenemos 2 botones los cuales son “Agregar” y “Editar”.

```
<div class="container card mb-5" style="width: 35rem;">
  <h1 class="display-6"><i class="fa fa-file-text-o" aria-hidden="true"></i> Formulario</h1>
  <div class="mt-3">
    <label for="id" class="form-label">Id</label>
    <input type="number" class="form-control" [(ngModel)]="formulario.id">
  </div>
  <div class="mt-3">
    <label for="nombre" class="form-label">Nombre</label>
    <input type="text" class="form-control" [(ngModel)]="formulario.nombre">
  </div>
  <div class="mt-3">
    <label for="precio" class="form-label">Precio</label>
    <input type="number" class="form-control" [(ngModel)]="formulario.precio">
  </div>
  <div class="mt-3 mb-3">
    <button class="btn btn-outline-success" (click)="agregar()">
      <i class="fa fa-plus-square-o" aria-hidden="true"></i> Agregar</button>
    <button class="btn btn-outline-warning" (click)="editar()" style="float: right;">
      <i class="fa fa-pencil-square-o" aria-hidden="true"></i> Editar</button>
  </div>
</div>
```

6.- En el apartado de los botones de igual manera por medio del método “click” nos va a ayudar a captar cuando el usuario presione el botón. En el primer boto el cual es “Agregar” solo va a ejecutar la función agregar() y no recibirá ningún elemento, como se muestra en el siguiente apartado de TS. El cual empieza con una validación por medio de una sentencia condicional “if” el cual solo verifica que ningún campo este vacío, después por medio de una variable nos vamos a apoyar para saber si el id ingresado existe o no.

```
public agregar() {
  if (this.formulario.id != null && this.formulario.nombre != null && this.formulario.precio != null && this.formulario.id > 0) {
    let existe = this.buscarRepetido(this.formulario.id);
    if (!existe) {
      let datoNuevo = {
        id: this.formulario.id,
        nombre: this.formulario.nombre,
        precio: this.formulario.precio
      };
      this.inventario.push(datoNuevo);
      this.limpiar();
      alert("Agregado con exito!!!");
    } else {
      alert("Ese Id ya existe, por favor agrega otro!!!");
    }
  } else {
    alert("Debes de llenar todos los campos!!!");
  }
}
```

Dentro de este método ejecutamos otros 2 métodos los cuales son limpiar() y bucarRepetido(), el primero solo limpia los campos a la hora que el usuario presione agregar y el segundo busca dentro del mismo si el id existe dentro de inventario.

```
public buscarRepetido(id: number) {
  let respuesta: boolean = false;
  for (let index = 0; index < this.inventario.length; index++) {
    if (this.inventario[index].id == id) {
      respuesta = true;
      break;
    }
  }
  return respuesta;
}
```

```
public limpiar() {
  this.formulario.id = null;
  this.formulario.nombre = null;
  this.formulario.precio = null;
}
```

7.- En el apartado del botón “Editar” de igual manera por medio del método “click” nos va a ayudar a captar cuando el usuario presione el botón. El cual por medio el método click solo recibirá la acción de la función editar(). Dentro de esta función hay una primera validación la cual se logra por medio de una sentencia condicional “if” en el cual se valida que los campos sean diferentes de null, después declaramos una variable id de tipo number la cual se le asignara lo que tenga formulario en el id. Después por una iteración vamos a leer todos los datos que tenga nuestro inventario, una vez que los lea por medio de otra sentencia condicional “if” vamos a preguntar si el id que se puso en el formulario es igual al del inventario para de esa forma poder realizar la actualización de datos, si no es así mandara el mensaje que debe de seleccionar un producto, de igual manera nos vamos a apoyar del método limpiar().

```
public editar() {  
  if (this.formulario.id != null && this.formulario.nombre != null && this.formulario.precio != null) {  
    let id: number = this.formulario.id;  
    for (let index = 0; index < this.inventario.length; index++) {  
      if (this.inventario[index].id == id) {  
        this.inventario[index].id = this.formulario.id;  
        this.inventario[index].nombre = this.formulario.nombre;  
        this.inventario[index].precio = this.formulario.precio;  
        this.limpiar();  
        alert("Se ha modificado con exito");  
      }  
    }  
  } else {  
    alert("Debes de seleccionar un producto")  
  }  
}
```


8.- Para poder darle mejor vista a nuestra maqueta instalaremos Bootstrap en angular por medio del siguiente comando, "npm install bootstrap@next" y una vez que ya se haya terminado de ejecutar nos iremos en el apartado de angular.json → "styles" y agregaremos la siguiente línea de comando, "./node_modules/bootstrap/dist/css/bootstrap.css".

```
"styles": [  
  "./node_modules/bootstrap/dist/css/bootstrap.css",  
  "node_modules/font-awesome/css/font-awesome.css",  
  "src/styles.css"  
],
```

9.- También para poder instalar Fontawesome en la terminal ingresaremos, "npm install font-awesome --save" y una vez que ya se haya terminado de ejecutar nos iremos en el apartado de angular.json → "styles" y agregaremos la siguiente línea de comando, "./node_modules/bootstrap/dist/css/bootstrap.css".

Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma <https://aka.ms/pscore6>

```
PS C:\Users\Admin\Documents\AngularProyectos\ngmodel> npm install font-awesome --save
```

10.- Ya por ultimo vamos a correr el comando "ng serve -o" para visualizar que en realidad se hayan puesto los cabios gráficos.

```
PS C:\Users\Admin\Documents\AngularProyectos\ngmodel> ng serve -o
✓ Browser application bundle generation complete.

Initial Chunk Files | Names          | Size
vendor.js           | vendor         | 2.68 MB
styles.css, styles.js | styles        | 607.00 kB
polyfills.js        | polyfills     | 508.81 kB
main.js             | main          | 24.52 kB
runtime.js          | runtime       | 6.57 kB

                | Initial Total | 3.80 MB

Build at: 2021-06-09T22:33:13.694Z - Hash: 5848cb9f8ca3dfae6731 - Time: 13314ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
```

Imagen de la página completa.



Imagen de la alerta de agregar campos vacíos.

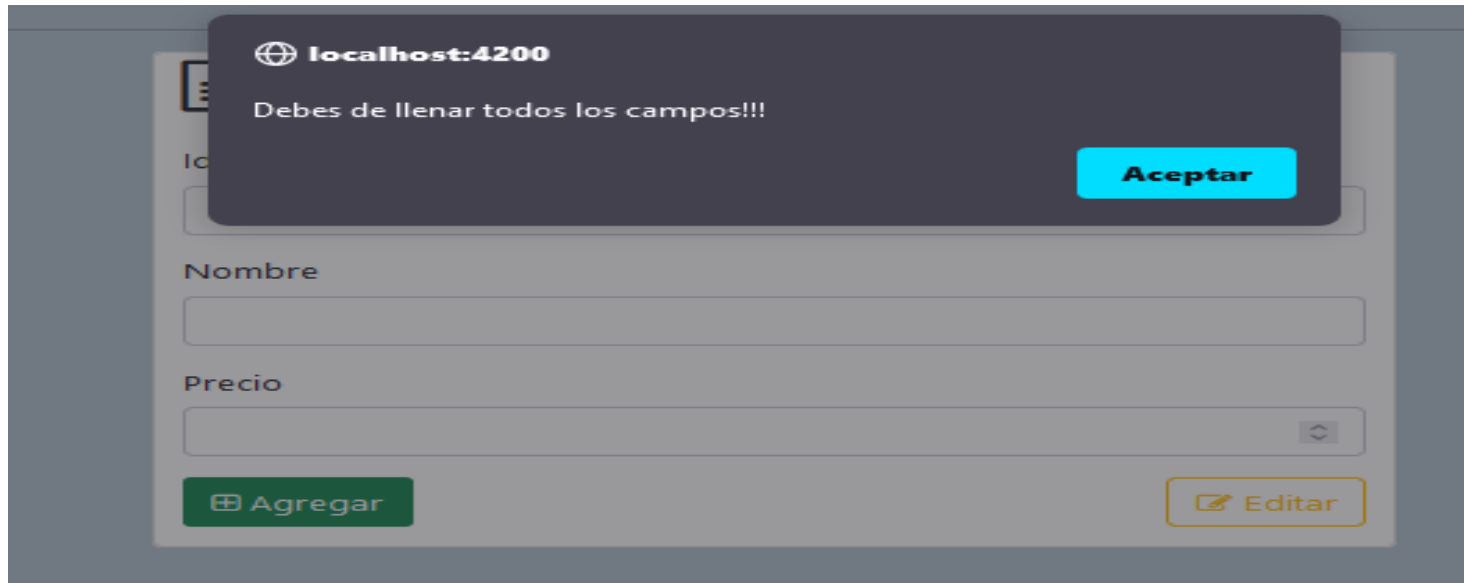


Imagen de la alerta de editar, debes de seleccionar un dato del inventario.

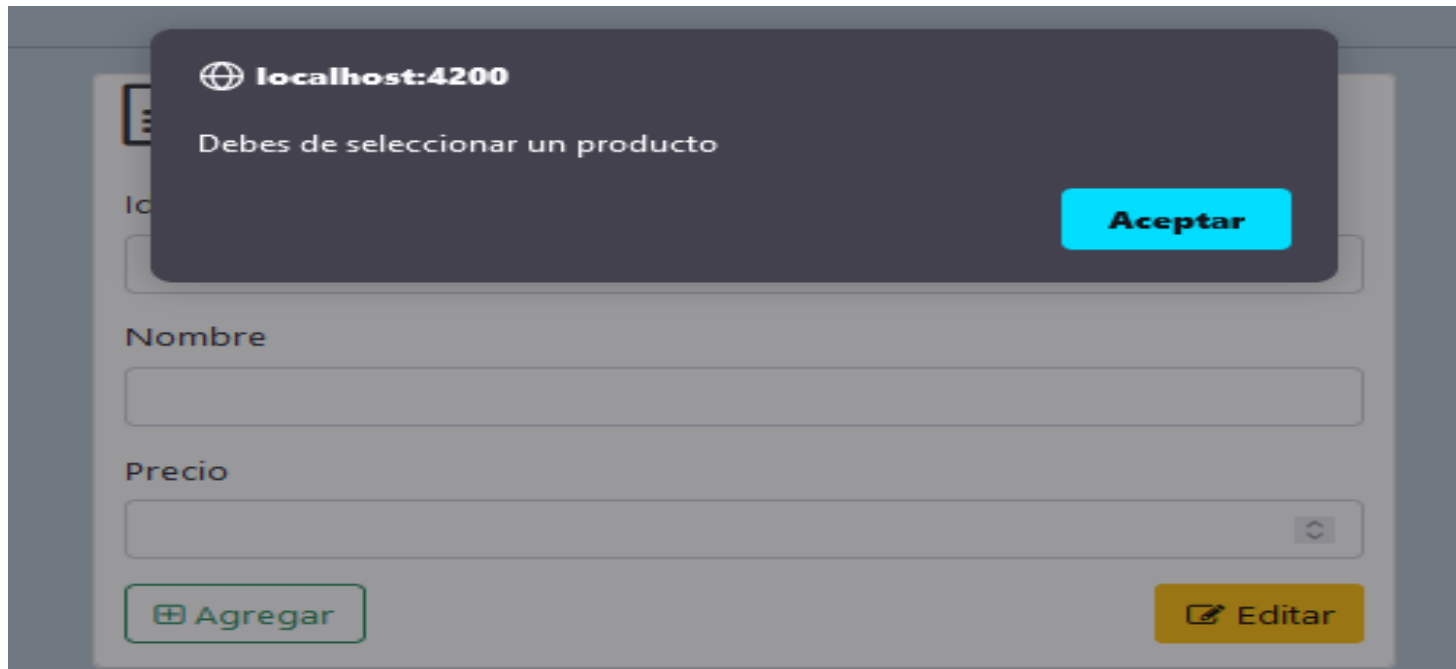


Imagen de la alerta cuando modificas un dato con éxito.

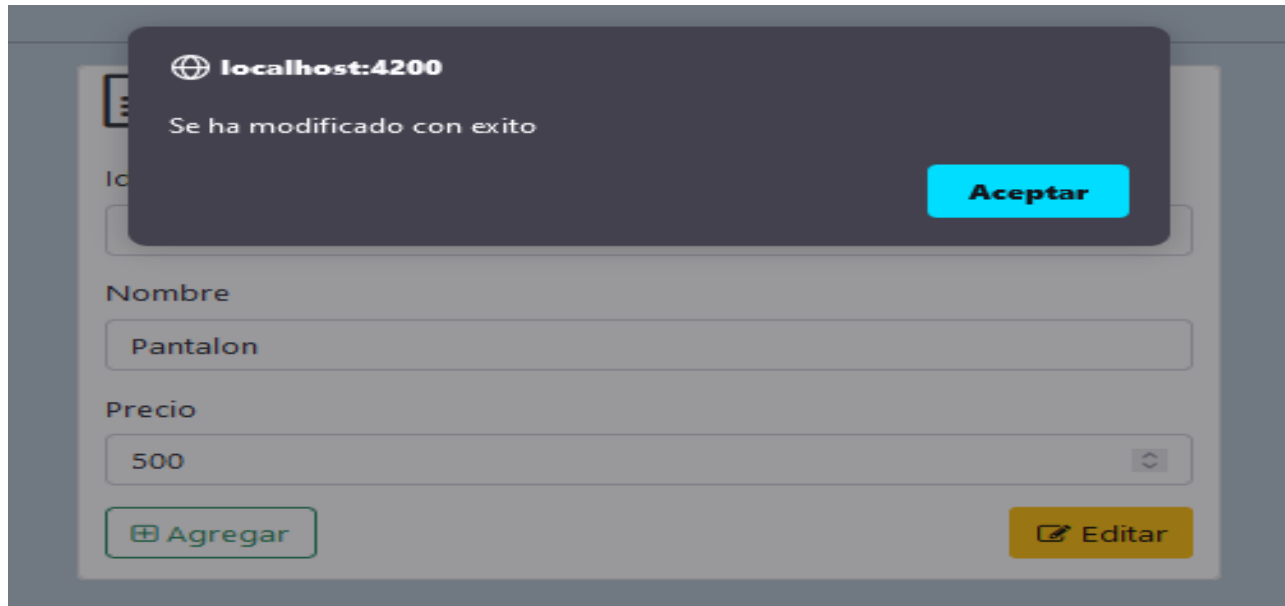
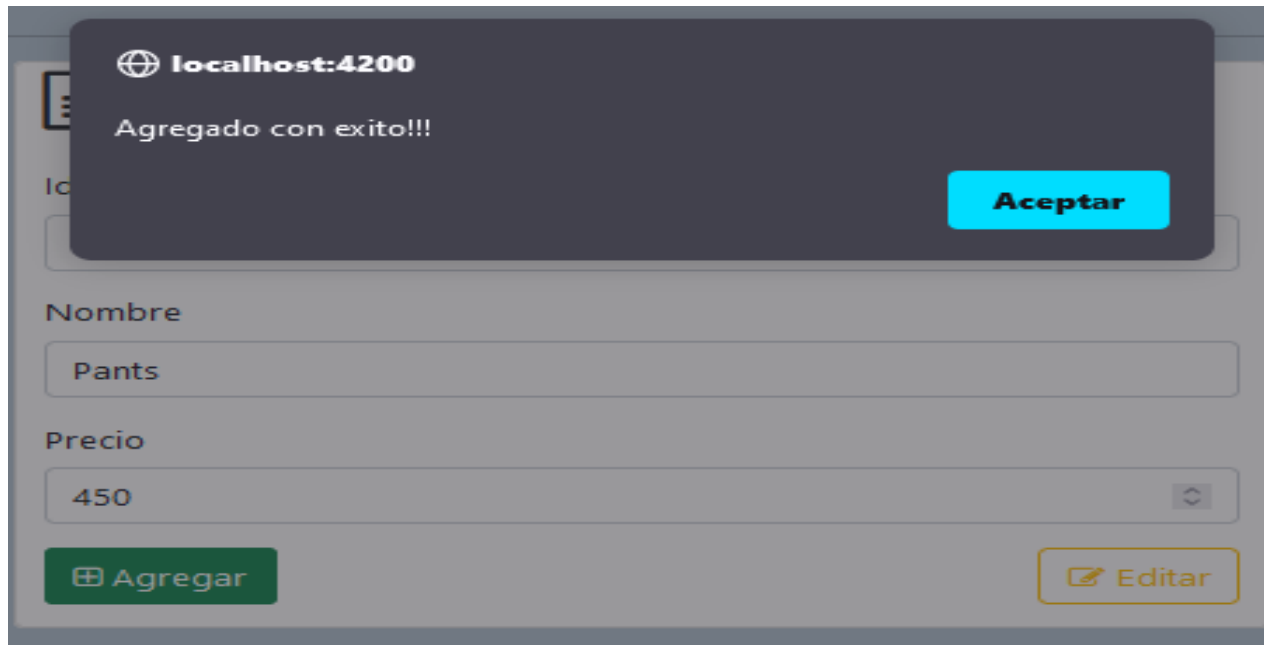




Imagen de la alerta cuando agregar un nuevo dato a inventario. Y de la tabla cuando ya se agrego el dato.



Id	Nombre	Precio	Borrar	Seleccionar
1	Pantalon	500	 Borrar	<input checked="" type="checkbox"/> Seleccionar
2	Chamarra	500	 Borrar	<input checked="" type="checkbox"/> Seleccionar
3	Pants	450	 Borrar	<input checked="" type="checkbox"/> Seleccionar

Por último, imagen de cuando borras un dato de la tabla. Ejemplo chamarra.



Id	Nombre	Precio	Borrar	Seleccionar
1	Pantalon	500	 Borrar	<input checked="" type="checkbox"/> Seleccionar
3	Pants	450	 Borrar	<input checked="" type="checkbox"/> Seleccionar