



UNIVERSIDAD POLITÉCNICA SALESIANA

MATERIA:

Programación y plataformas web

Proyecto Final

Realizado por:

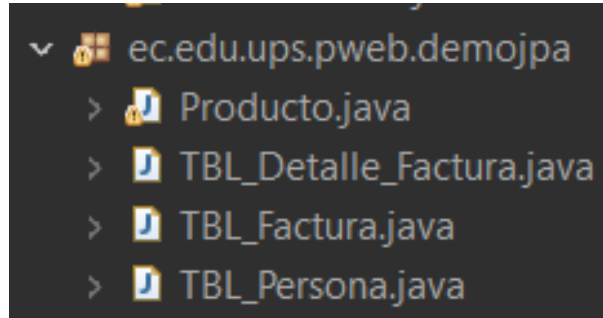
Santiago Torres

Andrés Alba

## Desarrollo

### Eclipse

- Para empezar en la parte de Backend debemos de definir nuestro modelo primeramente así que creamos nuestras clases Modelo donde las definimos en Entity, las variables que no pueden ser nulo y las conexiones con las otras tablas con el 'manytoOne'.



```
@Entity
public class TBL_Persona implements Serializable{
    private static final long serialVersionUID=1L ;
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "per_id")
    private int per_id;
    @Column(name = "per_cedula", unique=true)
    private String per_cedula;
    private String per_Nombre;
    private String per_Direccion;
    public int getPer_id() {
        return per_id;
    }
    public void setPer_id(int per_id) {
        this.per_id = per_id;
    }
    public String getPer_cedula() {
        return per_cedula;
    }
    public void setPer_cedula(String per_cedula) {
        this.per_cedula = per_cedula;
    }
    public String getPer_Nombre() {
        return per_Nombre;
    }
    public void setPer_Nombre(String per_Nombre) {
        this.per_Nombre = per_Nombre;
    }
    public String getPer_Direccion() {
        return per_Direccion;
    }
    public void setPer_Direccion(String per_Direccion) {
        this.per_Direccion = per_Direccion;
    }
}

@Entity
public class Producto implements Serializable{
    private static final long serialVersionUID = 1L;
    @Id
    @JoinColumn(name="pro_codigo")
    private int pro_codigo;
    private double pro_precio;
    private int pro_stock;
    private String pro_nombre;
    public int getPro_codigo() {
        return pro_codigo;
    }
    public void setPro_codigo(int pro_codigo) {
        this.pro_codigo = pro_codigo;
    }
    public double getPro_precio() {
        return pro_precio;
    }
    public void setPro_precio(double pro_precio) {
        this.pro_precio = pro_precio;
    }
    public int getPro_stock() {
        return pro_stock;
    }
    public void setPro_stock(int pro_stock) {
        this.pro_stock = pro_stock;
    }
    public String getPro_nombre() {
        return pro_nombre;
    }
    public void setPro_nombre(String pro_nombre) {
        this.pro_nombre = pro_nombre;
    }
}
```

```

@Entity
public class TBL_Factura implements Serializable{
    private static final long serialVersionUID=1L ;
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "fac_Codigo")
    private int fac_Codigo;
    private Integer per_id;
    @Temporal(TemporalType.DATE)
    private Date fac_fecha;
    private String activo;
    public Integer getPer_id() {
        return per_id;
    }
    public void setPer_id(Integer per_id) {
        this.per_id = per_id;
    }
    public Date getFac_fecha() {
        return fac_fecha;
    }
    public void setFac_fecha(Date fac_fecha) {
        this.fac_fecha = fac_fecha;
    }
    public Integer getFac_Codigo() {
        return fac_Codigo;
    }
}

public String getActivo() {
    return activo;
}
public void setActivo(String activo) {
    this.activo = activo;
}
@ManyToOne(cascade = {CascadeType.ALL})
@JoinColumn(name = "per_id", referencedColumnName = "per_id",
private TBL_Persona persona;

private int det_cantidad;
private double det_precio;

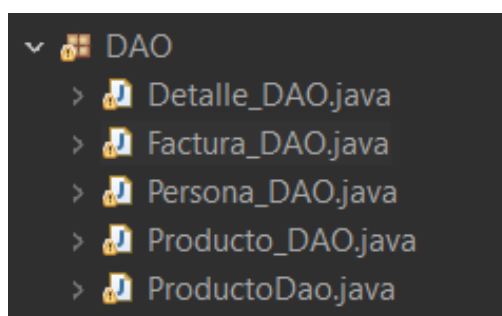
public int getFac_codigo() {
    return fac_codigo;
}
public void setFac_codigo(int fac_codigo) {
    this.fac_codigo = fac_codigo;
}
public int getPro_codigo() {
    return pro_codigo;
}
public void setPro_codigo(int pro_codigo) {
    this.pro_codigo = pro_codigo;
}
public int getDet_cantidad() {
    return det_cantidad;
}
public void setDet_cantidad(int det_cantidad) {
    this.det_cantidad = det_cantidad;
}
public double getDet_precio() {
    return det_precio;
}
public void setDet_precio(double det_precio) {
    this.det_precio = det_precio;
}

@ManyToOne(cascade = {CascadeType.ALL})
@JoinColumn(name = "fac_codigo",referencedColumnName = "fac_codigo", insertable =
private TBL_Factura factura ;

@ManyToOne(cascade = {CascadeType.ALL})
@JoinColumn(name = "pro_codigo",referencedColumnName = "pro_codigo", insertable =
private Producto producto ;

```

- Después de eso continuamos creando los DAO donde tendremos nuestro CRUD y usaremos el entity manager que es esencial para nuestro JPA ya que realiza los métodos de la conexión con la base de datos.



```

3 @Stateless
4 public class Persona_DAO implements Serializable{
5     private static final long serialVersionUID=1L ;
6     @PersistenceContext
7     private EntityManager ex;
8
9     public void crear(TBL_Persona persona) {
10         ex.persist(persona);
11     }
12     public TBL_Persona buscar(Integer codigo) {
13         TBL_Persona p= ex.find(TBL_Persona.class, codigo);
14         return p;
15     }
16
17     public void actualizar(TBL_Persona persona) {
18         ex.merge(persona);
19     }
20
21     public void eliminar(Integer codigo) {
22         TBL_Persona p=buscar(codigo);
23         try {
24             ex.remove(p);
25         }catch (Exception e) {
26             System.out.println("Veamos que onda");
27         }
28     }
29     public TBL_Persona buscarCedula(String cedula) {
30         String jsql="Select p from TBL_Persona p where p.per_cedula='"+cedula+"'";
31         Query query=ex.createQuery(jsql,TBL_Persona.class);
32         List<TBL_Persona> lista= query.getResultList();
33         return lista.get(0);
34     }
35     public List<TBL_Persona> listar() {
36         String jsql="SELECT p FROM TBL_Persona p";
37         Query query=ex.createQuery(jsql,TBL_Persona.class);
38         List<TBL_Persona> lista= query.getResultList();
39         return lista;
40     }
41 }

```

```

1 import ec.edu.ups.pweb.demojpa.Producto;
2 @Stateless
3 public class ProductoDao implements Serializable{
4     /**
5      *
6      */
7     private static final long serialVersionUID = 1L;
8     @PersistenceContext
9     private EntityManager em;
10
11     public void insert(Producto producto) {
12         em.persist(producto);
13     }
14
15     public Producto read(int codigo) {
16         Producto p= em.find(Producto.class, codigo);
17         return p;
18     }
19
20     public void update(Producto producto) {
21         em.merge(producto);
22     }
23
24     public void delete(int codigo) {
25
26         Producto p= em.find(Producto.class, codigo);
27         em.remove(p);
28     }
29
30     public List<Producto> getList(){
31         String jsql = "SELECT p FROM Producto p";
32         Query query = em.createQuery(jsql, Producto.class);
33         List<Producto> lista = query.getResultList();
34         return lista;
35     }
36 }

```

- Como nuestras facturas dependen de otras se usa una variable de Activo o Inactivo y para listarlas hacemos que nuestro Query a la base de datos tenga un where que haga que solo devuelva los que tenga T en su variable activo.

```

2
3 @Stateless
4 public class Factura_DAO implements Serializable {
5     private static final long serialVersionUID=1L ;
6     @PersistenceContext
7     private EntityManager ex;
8
9     public Integer crear(TBL_Factura factura) {
10         ex.persist(factura);
11         return factura.getFac_Codigo();
12     }
13     public TBL_Factura buscar(Integer codigo) {
14         TBL_Factura p= ex.find(TBL_Factura.class, codigo);
15         return p;
16     }
17     public void actualizar(TBL_Factura factura) {
18         ex.merge(factura);
19     }
20     public void eliminar(Integer codigo) {
21         TBL_Factura p=buscar(codigo);
22         ex.remove(p);
23     }
24
25     public List<TBL_Factura> listar() {
26         String jsq1="SELECT p FROM TBL_Factura p where p.activo='T'";
27         Query query=ex.createQuery(jsq1,TBL_Factura.class);
28         List<TBL_Factura> lista= query.getResultList();
29         return lista;
30     }
31 }

```

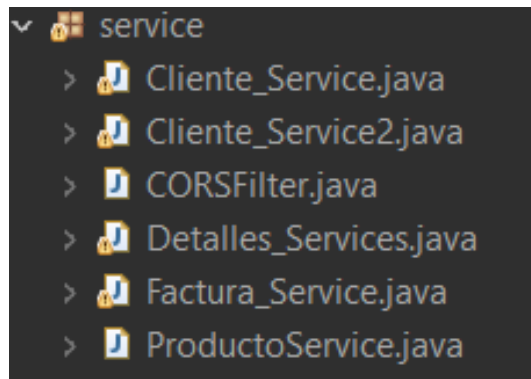
Y para listar nuestros detalles se hará con el código que reciba para realizar la búsqueda eso ya nos indica que se necesitará un método GET

```

2 @Stateless
3 public class Detalle_DAO implements Serializable {
4     private static final long serialVersionUID=1L ;
5     @PersistenceContext
6     private EntityManager ex;
7     public void crear(TBL_Detalle_Factura factura) {
8         ex.persist(factura);
9     }
10    public TBL_Detalle_Factura buscar(Integer codigo) {
11        TBL_Detalle_Factura p= ex.find(TBL_Detalle_Factura.class, codigo);
12        return p;
13    }
14    public void actualizar(TBL_Detalle_Factura factura) {
15        ex.merge(factura);
16    }
17    public void eliminar(int codigo) {
18        TBL_Detalle_Factura p=buscar(codigo);
19        ex.remove(p);
20    }
21
22    public List<TBL_Detalle_Factura> listar(Integer codigo) {
23        String jsq1="SELECT p FROM TBL_Detalle_Factura p where p.fac_codigo="+codigo;
24        Query query=ex.createQuery(jsq1,TBL_Detalle_Factura.class);
25        List<TBL_Detalle_Factura> lista= query.getResultList();
26        return lista;
27    }
28 }
29 }

```

- Ya solo nos falta los servicios que es por donde nos conectaremos con el Front-End donde tendremos Servicios de POST cuando tengamos que hacer una modificación en nuestra base de datos y un GET cuando necesitemos realizar consultas a la base de datos sabiendo que podemos usar el @QueryParam para obtener datos del Front-End



#### PersonaService

```
@Inject
private Persona_DAO persona ;

@GET
@Path("/buscaCliente")
@Produces(MediaType.APPLICATION_JSON)
public TBL_Persona buscaCedula(@QueryParam("cedula")String cedula) {
    return persona.buscarCedula(cedula);
}

@POST
@Produces(MediaType.APPLICATION_JSON)
@Consumes("application/json")
public String guardarPersonaObjeto(TBL_Persona p) {
    try {
        persona.eliminar(p.getPer_id());
    }catch (Exception e) {
        return "{\"Eliminar\": \"ERROR\"}";
    }
    return "{\"Eliminar\": \"OK\"}";
}

@POST
@Path("/actualizar")
@Produces(MediaType.APPLICATION_JSON)
@Consumes("application/json")
public String editarPersonaObjeto(TBL_Persona p) {
    try {
        persona.actualizar(p);
    }catch (Exception e) {
        return "{\"Editar\": \"ERROR\"}";
    }
    return "{\"Editar\": \"OK\"}";
}
```

```

@POST
@Produces(MediaType.APPLICATION_JSON)
@Consumes("application/json")
public String guardarPersonaObjeto(TBL_Persona p) {
    try {
        persona.crear(p);
    } catch (Exception e) {
        return "{\\"resultado\\": \\"ERROR\\"}";
    }
    return "{\\"resultado\\": \\"OK\\"}";
}

//CLIENTE SERVICES
@GET
@Path("/listado")
@Produces(MediaType.APPLICATION_JSON)
public List<TBL_Persona> getListadoPersonas(){
    return persona.listar();
}

```

### ProductoService

```

14
15 @Path("productos")
16 public class ProductoService {
17
18     @Inject
19     ProductoDao daoProducto;
20
21     @POST
22     @Path("newPro")
23     @Produces(MediaType.APPLICATION_JSON)
24     @Consumes("application/json")
25     public String guardarProducto(Producto p) {
26         try {
27             daoProducto.insert(p);
28         } catch (Exception e) {
29             return "{\\"resultado\\": \\"ERROR\\"}";
30         }
31         return "{\\"resultado\\": \\"OK\\"}";
32     }
33
34     @POST
35     @Path("findPro")
36     @Produces(MediaType.APPLICATION_JSON)
37     @Consumes("application/json")
38     public Producto buscarProducto(int codigo) {
39         return daoProducto.read(codigo);
40     }
41
42     @POST
43     @Path("updatePro")
44     @Produces(MediaType.APPLICATION_JSON)
45     @Consumes("application/json")
46     public String actualizarPro(Producto p) {
47         try {
48             daoProducto.update(p);
49         } catch (Exception e) {
50             return "{\\"resultado\\": \\"ERROR\\"}";
51         }
52         return "{\\"resultado\\": \\"OK\\"}";

```

```

42     @POST
43     @Path("updatePro")
44     @Produces(MediaType.APPLICATION_JSON)
45     @Consumes("application/json")
46     public String actualizarPro(Producto p) {
47         try {
48             daoProducto.update(p);
49         } catch (Exception e) {
50             return "{\"resultado\": \"ERROR\"}";
51         }
52         return "{\"resultado\": \"OK\"}";
53     }
54
55     }
56     @POST
57     @Path("deletePro")
58     @Produces(MediaType.APPLICATION_JSON)
59     @Consumes("application/json")
60     public String eliminarPro(int codigo) {
61         try {
62             System.out.println(codigo);
63             daoProducto.delete(codigo);
64         } catch (Exception e) {
65             return "{\"resultado\": \"ERROR\"}";
66         }
67         return "{\"resultado\": \"OK\"}";
68     }
69
70     }
71
72     @GET
73     @Path("listPro")
74     @Produces(MediaType.APPLICATION_JSON)
75     public List<Producto> getListPer(){
76         return daoProducto.getList();
77     }
78     //Content-type=application/json
79 }
80

```

FacturaService



```

@Path("/facturas")
public class Factura_Service {
    @Inject
    private Factura_DAO daoFactura_DAO;

    @GET
    @Path("/listar")
    @Consumes("application/json")
    @Produces(MediaType.APPLICATION_JSON)
    public List<TBL_Factura> getFactura(){
        return daoFactura_DAO.listar();
    }

    @POST
    @Path("/crear")
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes("application/json")
    public Integer crearFactura(TBL_Factura p) {
        Integer codigo=-1;
        try {
            codigo=daoFactura_DAO.crear(p);
        }catch (Exception e) {
            return codigo;
        }
        return codigo;
    }

    @POST
    @Path("/anular")
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes("application/json")
    public String anularFactura(Integer codigo) {
        System.out.println("llego"+codigo);
        TBL_Factura p=daoFactura_DAO.buscar(codigo);
        System.out.println(p.getFac_Codigo()+"--"+p.getActivo());
        p.setActivo("F");
        try {
            daoFactura_DAO.actualizar(p);
        }
    }
}

```

### DetallesServices

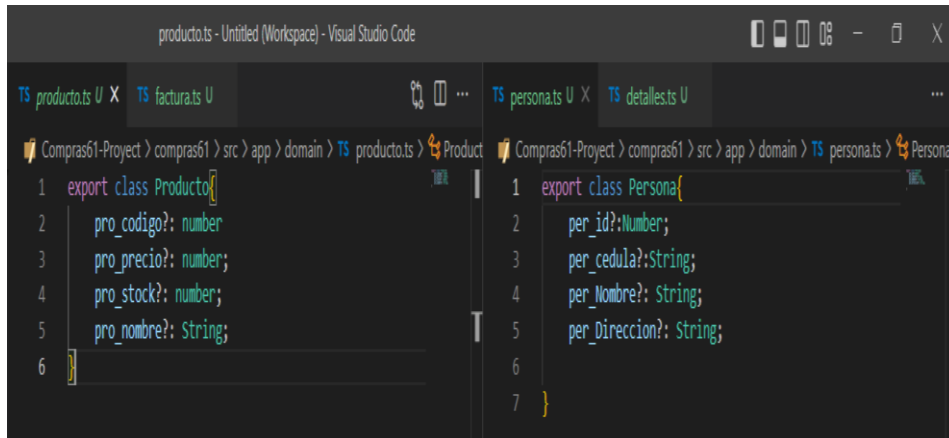
```

9 @Path("/detalles")
0 public class Detalles_Services {
1
2     @Inject
3     private Detalle_DAO detalle ;
4
5     @GET
6     @Path("/listar")
7     @Produces(MediaType.APPLICATION_JSON)
8     public List<TBL_Detalle_Factura> getListadoPersonas(@QueryParam("codigo")Integer codigo){
9         return detalle.listar(codigo);
0     }
1
2     @POST
3     @Path("/creaDetalles")
4     @Produces(MediaType.APPLICATION_JSON)
5     @Consumes("application/json")
6     public String guardarProducto(TBL_Detalle_Factura p) {
7         try {
8             detalle.crear(p);
9         }catch (Exception e){
0             return "{\"resultado\": \"ERROR\"}";
1         }
2         return "{\"resultado\": \"OK\"}";
3     }
4 }
5
6

```

## Visual Studio Code (Angular)

- Comenzamos creando los objetos con las mismas variables que tenemos en el Eclipse.

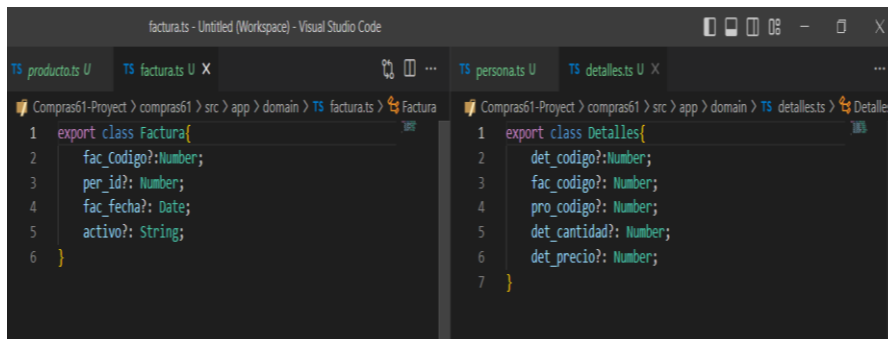


The screenshot shows the Visual Studio Code interface with two files open: `producto.ts` and `persona.ts`. The `producto.ts` file contains the following code:

```
1 export class Producto{
2     pro_codigo?: number;
3     pro_precio?: number;
4     pro_stock?: number;
5     pro_nombre?: String;
6 }
```

The `persona.ts` file contains the following code:

```
1 export class Persona{
2     per_id?: Number;
3     per_cedula?: String;
4     per_nombre?: String;
5     per_direccion?: String;
6 }
```



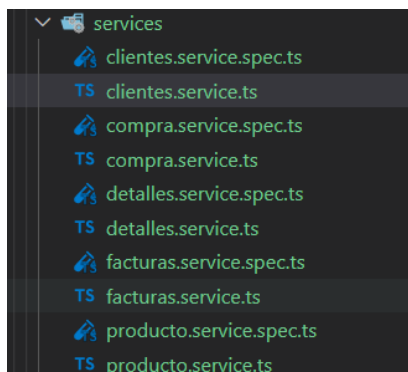
The screenshot shows the Visual Studio Code interface with two files open: `factura.ts` and `detalles.ts`. The `factura.ts` file contains the following code:

```
1 export class Factura{
2     fac_codigo?: Number;
3     per_id?: Number;
4     fac_fecha?: Date;
5     activo?: String;
6 }
```

The `detalles.ts` file contains the following code:

```
1 export class Detalles{
2     det_codigo?: Number;
3     fac_codigo?: Number;
4     pro_codigo?: Number;
5     det_cantidad?: Number;
6     det_precio?: Number;
7 }
```

- Luego creamos los services que utilizaremos para llamar a los métodos GET y POST que se encuentran en el Eclipse, pasando los respectivos parámetros.



```

// clientes.service.ts
11 constructor(private http: HttpClient) {
12 }
13
14 save(persona: Persona): Observable<Persona> {
15   return this.http.post<Persona>("http://localhost:8080/demo/jpa/r
16 }
17
18 obtener(): Observable<any[]>{
19   return this.http.get<any>("http://localhost:8080/demo/jpa/rs/cli
20 }
21
22 eliminar(persona: Persona): Observable<Persona> {
23   return this.http.post<Persona>("http://localhost:8080/demo/jpa/r
24 }
25
26 editar(p: Persona): Observable<any[]>{
27   return this.http.post<any>("http://localhost:8080/demo/jpa/rs/cl
28 }
29
30 buscar(p:string): Observable<any>{
31   console.log(p)
32   return this.http.get<any>("http://localhost:8080/demo/jpa/rs/cli
33 }
34
35 }
36
37
// clientes.service.ts
8 providedIn: 'root'
9 })
10 export class ClientesService {
11
12   constructor(private http: HttpClient) {
13
14   }
15
16   save(persona: Persona): Observable<Persona> {
17     return this.http.post<Persona>("http://localhost:8080/demo/jpa/r
18   }
19
20   obtener(): Observable<any[]>{
21     return this.http.get<any>("http://localhost:8080/demo/jpa/rs/cli
22   }
23
24   eliminar(persona: Persona): Observable<Persona> {
25     return this.http.post<Persona>("http://localhost:8080/demo/jpa/r
26   }
27
28   editar(p: Persona): Observable<any[]>{
29     return this.http.post<any>("http://localhost:8080/demo/jpa/rs/cl
30   }
31   buscar(p:string): Observable<any>{
32     console.log(p)
33     return this.http.get<any>("http://localhost:8080/demo/jpa/rs/cli
34   }
35
36 }
37

```

- Continuaremos con la creación de cada uno de los componentes que utilizaremos para crear, listar o editar cada uno de los objetos. Donde en cada archivo ts tenemos la codificación que nos permite llamar a los metodos para mostrar en el html y así mismo mandar datos desde el Frontend al Backend.



```
ponent.ts - Untitled (Workspace) - Visual Studio Code

TS pro-listar.component.ts U X

Compras61-Proyect > compras61 > src > app > Power > Producto > pro-listar > TS pro-listar.component.ts > ...

24
25   ngOnInit(): void {
26   }
27   this.listar()
28 }
29 guardar(){
30   this.productoService.guardar(this.producto).subscribe(data=>{
31     console.log(data);
32   })
33 }
34   this.producto = new Producto()
35 }
36 }
37
38 listar(){
39
40   this.lstProductos = this.productoService.listar().subscribe(
41     data => this.lstProductos = data,
42     error => console.log(error)
43   );
44 }
45 }
46 editProducto(p: Producto){
47
48   let params: NavigationExtras = {
49     queryParams: {
50       codigo: p.pro_codigo,
51       precio: p.pro_precio,
52       stock: p.pro_stock,
53       nombre: p.pro_nombre
54     }
55   }
```

```
pro-editor.component.ts - Untitled (Workspace) - Visual Studio Code

TS pro-editor.component.ts U X

Compras61-Proyect > compras61 > src > app > Power > Producto > pro-editor > TS pro-editor.component.ts > ...

8   templateUrl: './pro-editor.component.html',
9   styleUrls: ['./pro-editor.component.scss']
10 })
11 export class ProEditorComponent implements OnInit {
12   p:Producto=new Producto()
13   constructor(private router: Router,
14     private productoService: ProductoService) {
15     let params = this.router.getCurrentNavigation()?.extras.queryParams;
16     if(params){
17       this.p = new Producto()
18       this.p.pro_codigo = params['codigo'];
19       this.p.pro_precio = params['precio'];
20       this.p.pro_stock = params['stock'];
21       this.p.pro_nombre = params['nombre'];
22     }
23   }
24
25   ngOnInit(): void {
26   }
27   edit(){
28     if(this.p.pro_nombre!=null && this.p.pro_precio!=null && this.p.pro_stock!=null){
29       this.productoService.editar(this.p).subscribe(data=>{
30         console.log(data);
31       })
32       this.router.navigate(['productos/listPro'])
33     }else{
34       alert("Porfavor rellene todos los campos")
35     }
36   }
37 }
38
39   getListClientes()
```

```

8      styleUrls: ['./pro-crear.component.scss']
9    })
10   export class ProCrearComponent implements OnInit {
11
12
13     producto = new Producto()
14     constructor(private productService: ProductService,
15                 private router: Router) {
16       this.producto = new Producto()
17     }
18
19     ngOnInit(): void {
20
21     }
22     guardar(){
23       if(this.producto.pro_codigo!=null && this.producto.pro_nombre!=null && this.producto.pro_precio!=null && this.producto.pro_fecha!=null){
24         this.productoService.guardar(this.producto).subscribe((data:Object )=>{
25
26           if((Object.entries(data)).toString() == "resultado,ERROR"){
27             alert("El objeto no se ha ingresado correctamente")
28           }else{
29             alert("Producto registrado correctamente")
30             this.producto = new Producto()
31           }
32         })
33       }else{
34         alert("Porfavor rellene todos los campos")
35       }
36     }
37
38   }

```

- Después podemos seguir con la creación del HTML con su respectivo CSS para lograr una presentación aceptable a la vista.

```
pro-listar.component.html - Untitled (Workspace) - Visual Studio Code  
Go to component:  
<header id="cabecera">  
    
  <a id="logo">  
    <span class="TIT">Universidad Politécnica Salesiana</span>  
    <span class="SUBTIT">PROYECTO FINAL PROGRAMACIÓN WEB</span>  
  </a><!-- / #logo-header -->  
  <nav>  
    <ul>  
      <li (click)="goListClientes()"><a>Gestionar Clientes</a></li>  
      <li (click)="goToIst()"><a>Gestionar Productos</a></li>  
      <li (click)="goListaFac()"><a>Gestionar Facturas</a></li>  
      <li (click)="goCreaCompra()"><a>Realizar Compra</a></li>  
    </ul>  
  </nav><!-- / nav -->  
</header>  
<body>  
  <br><br><br>  
  <table rules=all frame="box">  
    <tr id="cabecera_tabla">  
      <th id="txtNegrita">&nbsp;&nbsp;&nbsp;&Codigo&nbsp;&nbsp;&nbsp;</th>  
      <th id="txtNegrita">&nbsp;&nbsp;&Nombre&nbsp;&nbsp;&nbsp;</th>  
      <th id="txtNegrita">&nbsp;&Precio&nbsp;&nbsp;&nbsp;</th>  
      <th id="txtNegrita">&nbsp;&Stock&nbsp;&nbsp;&nbsp;</th>  
      <th id="txtNegrita">&nbsp;&Acciones</th>  
    </tr>  
  
    <tr *ngFor="let p of lstProductos" >  
      <th id="txtNoNegrita">{{p.pro_codigo}}</th>  
      <th id="txtNoNegrita">{{p.pro_nombre}}</th>  
      <th id="txtNoNegrita">{{p.pro_precio}}</th>  
      <th id="txtNoNegrita">{{p.pro_stock}}</th>  
      <th colspan="2">&nbsp;&nbsp;&mat-raised-button color="accent"  
        &nbsp;&nbsp;&mat-raised-button color="accent">Elimi
```

```
pro-listar.component.scss - Untitled (Workspace) - Visual Studio Code

pro-listar.component.html U  pro-listar.component.scss U X

Compras61-Proyect > compras61 > src > app > Power > Producto > pro-listar > pro-listar.component.scss > html

1  html {
2    height: 100%;
3  }
4  body {
5    min-height: 100%;
6  }
7  #cabecera {
8    background: #333;
9    font-size: medium;
10   height: 80px;
11   width: 100%;
12   left: 0;
13   top: 0;
14   position: fixed;
15   a {
16     color: #52b2fc;
17   }
18 }
19 #logo {
20   float: left;
21   padding: 15px 0 0 20px;
22   text-decoration: none;
23   &:hover {
24     color: #0b76a6;
25   }
26   .TIT {
27     display: block;
28     font-weight: 700;
29     font-size: 1.2em;
30     color: #0b76a6;
31   }
32   .SUBTIT {
33     display: block;
34     font-weight: 300;
35     font-size: 0.8em;
36     color: #999;
37   }
38 }
```

```
pro-editar.component.html - Untitled (Workspace) - Visual Studio Code

pro-editar.component.html U X

Compras61-Proyect > compras61 > src > app > Power > Producto > pro-editar > pro-editar.component.html > header#cabecera

Go to component
1  <header id="cabecera">
2    
3    <a id="logo">
4      <span class="TIT">Universidad Politécnica Salesiana</span>
5      <span class="SUBTIT">PROYECTO FINAL PROGRAMACIÓN WEB</span>
6    </a> <!-- / #logo-header -->
7    <nav>
8      <ul>
9        <li (click)="goListClientes()"><a>Gestionar Clientes</a></li>
10       <li (click)="goToList()"><a>Gestionar Productos</a></li>
11       <li (click)="goListaFac()"><a>Gestionar Facturas</a></li>
12       <li (click)="goCreaCompra()"><a>Realizar Compra</a></li>
13     </ul>
14   </nav><!-- / nav -->
15
16 </header>
17 <br><br><br><br>
18 <div >
19   <div class="center">
20     <form>
21
22       <label id="txtNegrita" for="txtCode">Codigo&nbsp;&nbsp;&nbsp;</label><br>
23       <input id="txtCode" name="txtCode" readonly="readonly" [(ngModel)]="p.pro_codigo" disabled="true"/><br>
24       <label id="txtNegrita" for="txtName">Nombre&nbsp;&nbsp;&nbsp;</label><br>
25       <input id="txtName" name="txtName" [(ngModel)]="p.pro_nombre"/><br>
26       <label id="txtNegrita" for="txtName">Precio&nbsp;&nbsp;&nbsp;</label><br>
27       <input type="number" id="a" name="a" [(ngModel)]="p.pro_precio"/><br>
28       <label id="txtNegrita" for="txtPrice">Stock&nbsp;&nbsp;&nbsp;</label><br>
29       <input type="number" id="txtPrice" name="txtPrice" [(ngModel)]="p.pro_stock"/><br>
30
31       <button id="miBotonBonito" (click)="edit()" mat-raised-button color="accent">Guardar</button>
32
33     </form>
34   </div>
35 </div>
36
```

```
pro-editar.component.scss - Untitled (Workspace) - Visual Studio Code
pro-editar.component.scss U x
Compras61-Proyect > compras61 > src > app > Power > Producto > pro-editar > pro-editar.component.scss > #logo > &:hover
1 #cabecera {
2   background: #333;
3   font-size: medium;
4   height: 80px;
5   width: 100%;
6   left: 0;
7   top: 0;
8   position: fixed;
9   a {
10    color: #52b2fc;
11  }
12 }
13
14 #logo {
15   float: left;
16   padding: 15px 0 0 20px;
17   text-decoration: none;
18   &:hover {
19     color: #0b76a6;
20   }
21 }
22 .TIT {
23   display: block;
24   font-weight: 700;
25   font-size: 1.2em;
26   color: #0b76a6;
27 }
28 .SUBTIT {
29   display: block;
30   font-weight: 300;
31   font-size: 0.8em;
32   color: #999;
33 }
34 nav {
35   float: right;
36   ul {
37     margin: 0;
38     padding: 0;
```

- Dando así el siguiente resultado

