

**UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA  
FACULTAD DE INGENIERIA EN SISTEMAS DE INFORMACIÓN  
LICENCIATURA EN INGENIERÍA EN SISTEMAS DE INFORMACIÓN  
CENTRO UNIVERSITARIO DE CHIMALTENANGO**

**CUARTO CICLO  
PROGRAMACION II**



**MANUAL TECNICO DE LA APLICACIÓN WEB**

**Daniel Angel Ambrocio Coj 1990-22-13443**

-----  
**PROGRAMADOR**

**CHIMALTENANGO, NOVIEMBRE DE 2023**

## **Tecnologías Utilizadas**

La integración de Apache NetBeans IDE 19 con Spring Boot y las tecnologías web proporciona un entorno de desarrollo robusto y eficiente. NetBeans simplifica la creación y gestión de proyectos Java, mientras que Spring Boot facilita el desarrollo de aplicaciones Java empresariales. La combinación de HTML, CSS y JavaScript permite la creación de interfaces de usuario atractivas y dinámicas, brindando una solución integral para el desarrollo ágil y efectivo de aplicaciones Java modernas.

### **Backend con Spring Boot y Java:**

- **Spring Boot:** Facilita el desarrollo del backend al proporcionar un marco de trabajo simplificado y basado en convenciones. Ofrece características como la configuración automática, lo que reduce la carga de trabajo y acelera el desarrollo.
- **Java:** Al ser un lenguaje de programación versátil y robusto, Java se integra perfectamente con Spring Boot para la implementación del backend. Proporciona rendimiento, seguridad y escalabilidad.

### **Frontend con HTML, CSS y JavaScript:**

- **HTML (Hypertext Markup Language):** Esencial para la estructura básica de la interfaz de usuario. Permite definir la jerarquía y organización de los elementos en la página web.
- **CSS (Cascading Style Sheets):** Facilita la presentación y el diseño de la interfaz de usuario. Permite personalizar el aspecto visual, mejorar la usabilidad y garantizar la coherencia en todo el sitio.

- **JavaScript:** Agrega dinamismo y funcionalidades interactivas a la interfaz de usuario. Facilita la manipulación del DOM (Modelo de Objetos del Documento) y permite realizar operaciones asíncronas para mejorar la experiencia del usuario.

#### **Entorno de Desarrollo con Apache NetBeans IDE 19:**

- **Integración de Herramientas:** NetBeans simplifica el desarrollo al proporcionar herramientas integradas para la creación, prueba y depuración de código. Su compatibilidad con Spring Boot facilita la gestión de proyectos y la implementación del backend.
- **Desarrollo Visual del Frontend:** La capacidad de edición de HTML, CSS y JavaScript en NetBeans permite una experiencia de desarrollo visual y eficiente para el frontend de la aplicación.

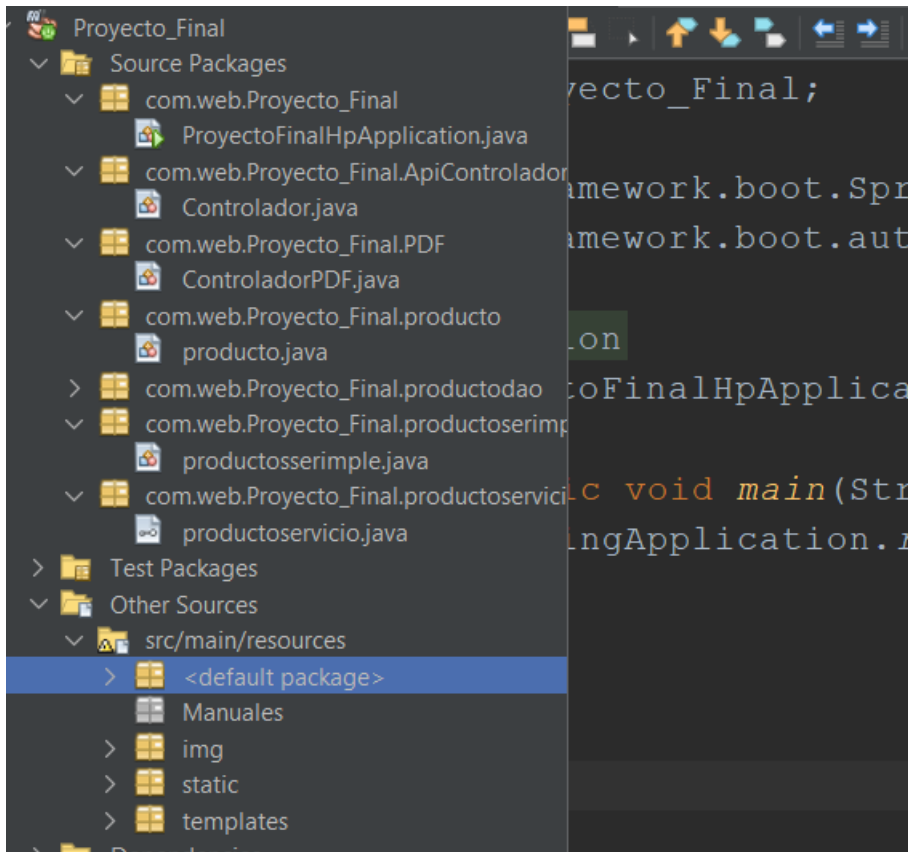
#### **Ventajas Globales:**

- **Productividad:** La integración de estas tecnologías permite una mayor productividad al simplificar tareas y proporcionar herramientas eficientes para el desarrollo.
- **Escalabilidad:** La combinación de Spring Boot y Java ofrece una base sólida para el backend, brindando escalabilidad a medida que la aplicación crece.
- **Experiencia de Usuario Mejorada:** El uso de HTML, CSS y JavaScript en el frontend permite la creación de interfaces atractivas y dinámicas, mejorando la experiencia del usuario.

## Backend

Al realizar el backen con java y spring boot, se tiene la facilidad de ahorrarse líneas de código ya que spring boot se encarga de eso.

El proyecto consta una disertes paquete que funcionan simultanea mente para realizar el CRUD.

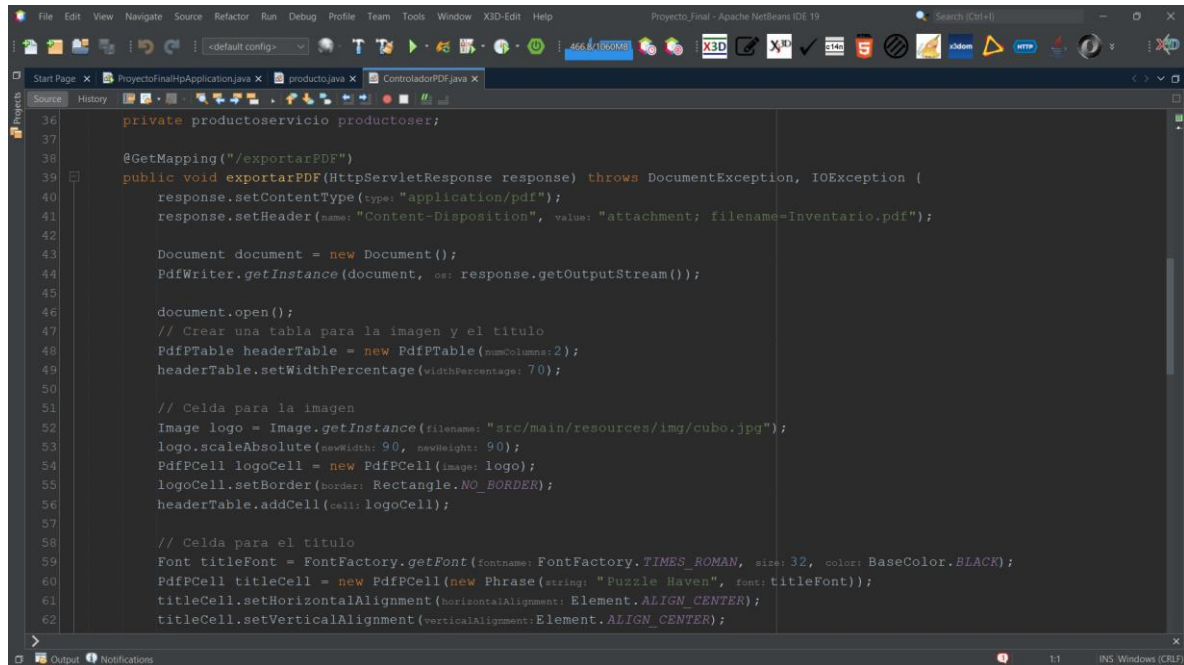


## Controlador

Para realizar la aplicación web se necesita de un controlador, el controlador es lo esencial y en la parte donde va toda la lógica al manejar los diferentes archivos html.

### Controlador de PDF

Utilizados un controlador aparte para poder exportar los productos del inventario a un PDF.



```
36 private productoservicio productoserv;
37
38 @GetMapping("/exportarPDF")
39 public void exportarPDF(HttpServletResponse response) throws DocumentException, IOException {
40     response.setContentType("application/pdf");
41     response.setHeader("Content-Disposition", "attachment; filename=Inventario.pdf");
42
43     Document document = new Document();
44     PdfWriter.getInstance(document, os: response.getOutputStream());
45
46     document.open();
47     // Crear una tabla para la imagen y el titulo
48     PdfPTable headerTable = new PdfPTable(numColumns:2);
49     headerTable.setWidthPercentage(widthPercentage: 70);
50
51     // Celda para la imagen
52     Image logo = Image.getInstance(filename: "src/main/resources/img/cubo.jpg");
53     logo.scaleAbsolute(newWidth: 90, newHeight: 90);
54     PdfPCell logoCell = new PdfPCell(image: logo);
55     logoCell.setBorder(border: Rectangle.NO_BORDER);
56     headerTable.addCell(cell: logoCell);
57
58     // Celda para el titulo
59     Font titleFont = FontFactory.getFont(fontname: FontFactory.TIMES_ROMAN, size: 32, color: BaseColor.BLACK);
60     PdfPCell titleCell = new PdfPCell(new Phrase(string: "Puzzle Haven", font: titleFont));
61     titleCell.setHorizontalAlignment(horizontalAlignment: Element.ALIGN_CENTER);
62     titleCell.setVerticalAlignment(verticalAlignment: Element.ALIGN_CENTER);
```

## Producto

Usamos un java class para preparar los campos que tiene la base de datos y prepara algunos get y set que se utilizaran.

```
@Column(name = "nombreProducto")
private String Nombre_Producto;

@Min(1)
private float Precio_Unitario;

@Min(1)
private int Cantidad_Producto;

@NotEmpty
private String Edad_de_Uso;

    public Long getCodigo_Producto() {
        return Codigo_Producto;
    }

    public void setCodigo_Producto(Long codigo_Producto) {
        Codigo_Producto = codigo_Producto;
    }

    public String getNombre_Producto() {
        return Nombre_Producto;
    }
```

## Productodao

En esta parte en lugar de usar una clase java se utilizó una interface y se extendió CRUD.

El programador solo tuvo que escribir unas líneas de código.

```
* Click nbfs://nbhost/SystemFileSystem/Templates/springboot/ApiRunner.java
*/
package com.web.Proyecto_Final.productodao;

import com.web.Proyecto_Final.producto.producto;
import org.springframework.data.repository.CrudRepository;

public interface productodao extends CrudRepository<producto, Long> {

}

}
```

Mientras que spring boot realizó más código automáticamente al extender el CRUD.

```
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package org.springframework.data.repository;

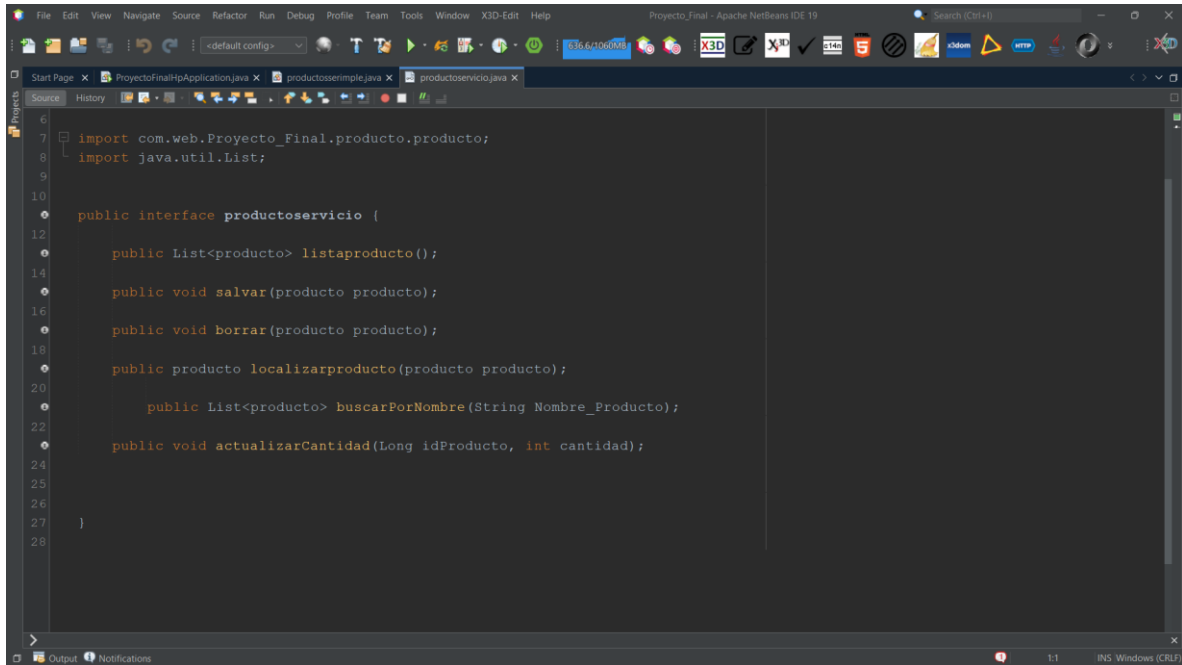
import java.util.Optional;
import org.springframework.dao.OptimisticLockingFailureException;

/**
 * Interface for generic CRUD operations on a repository for a specific type.
 *
 * @author Oliver Gierke
 * @author Eberhard Wolff
 * @author Jens Schauder
 */
@NoRepositoryBean
public interface CrudRepository<T, ID> extends Repository<T, ID> {

    /**
     * Saves a given entity. Use the returned instance for further operations as the
     * entity instance completely.
     *
     * @param entity must not be {@literal null}.
     */
}
```

## Productoservicio

El propósito de crear un productoservicio en Java con Spring Boot es aprovechar las ventajas específicas de esta tecnología para lograr un desarrollo eficiente, escalable, seguro y fácil de mantener, cumpliendo así con las necesidades del mercado y los usuarios.



```
6
7 import com.web.Proyecto_Final.producto.producto;
8 import java.util.List;
9
10
11
12 public interface productoservicio {
13
14     public List<producto> listaproducto();
15
16     public void salvar(producto producto);
17
18     public void borrar(producto producto);
19
20     public producto localizarproducto(producto producto);
21
22     public List<producto> buscarPorNombre(String Nombre_Producto);
23
24     public void actualizarCantidad(Long idProducto, int cantidad);
25
26
27 }
28
```



## Productoimplemento

En el producto implemento es necesario ya que implementa, lo que se realizó en productos y productoservicio.

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class productoserimple implements productoservicio{

    @Autowired
    private productodao productodaso;

    @Override
    @Transactional(readOnly = true)
    public List<producto> listaproducto() {
        return (List<producto>) productodaso.findAll();
    }

    @Override
    @Transactional
    public void salvar(producto producto) {
        productodaso.save(entity: producto);
    }

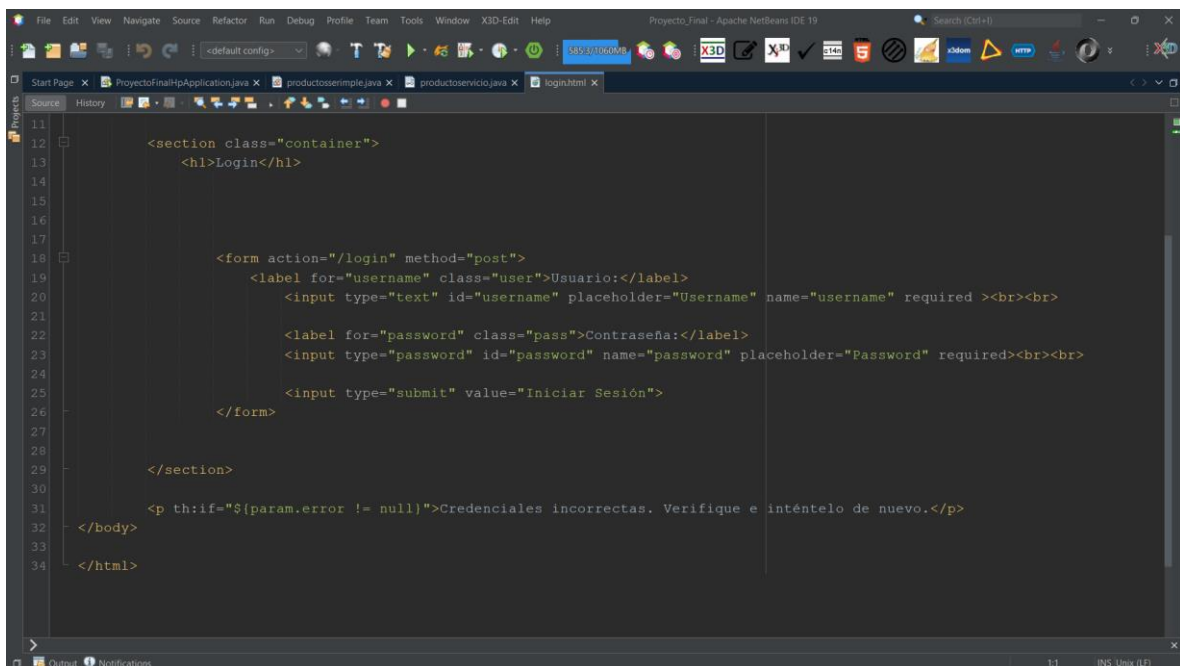
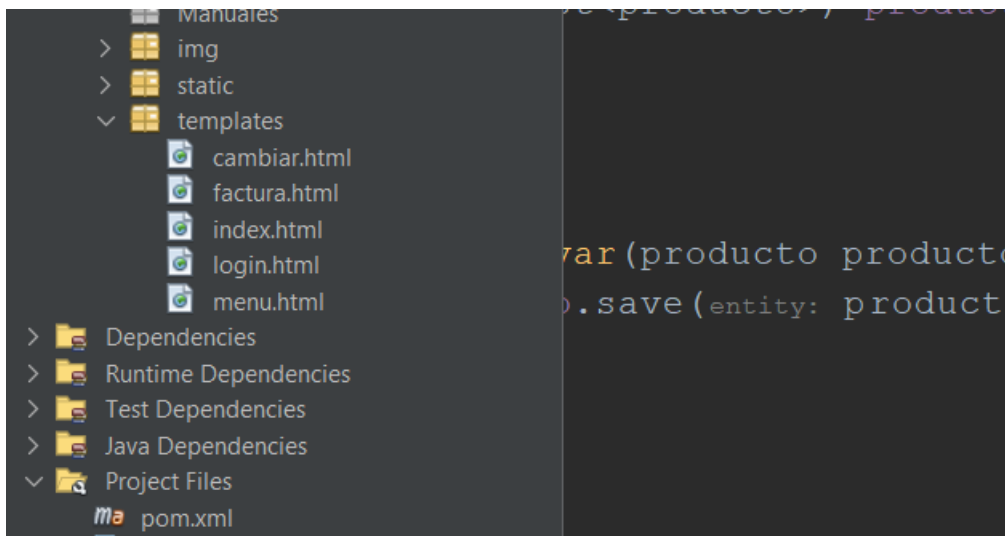
    @Override
    @Transactional
    public void borrar(producto producto) {
        productodaso.delete(entity: producto);
    }
}
```

## Frontend

Para realizar el frontend se utilizó html, css y java.

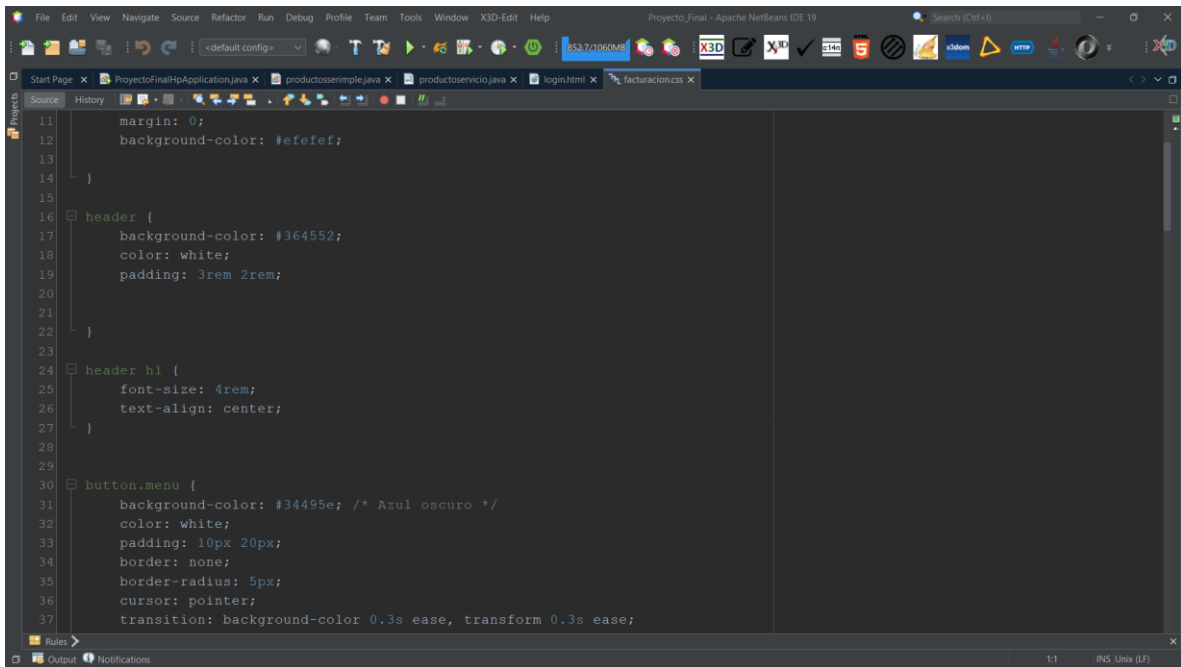
## Página

Los diferentes archivos html que se utilizaron para realizar el proyecto, fueron para crear una estructura de las diferentes funciones que posee la web.

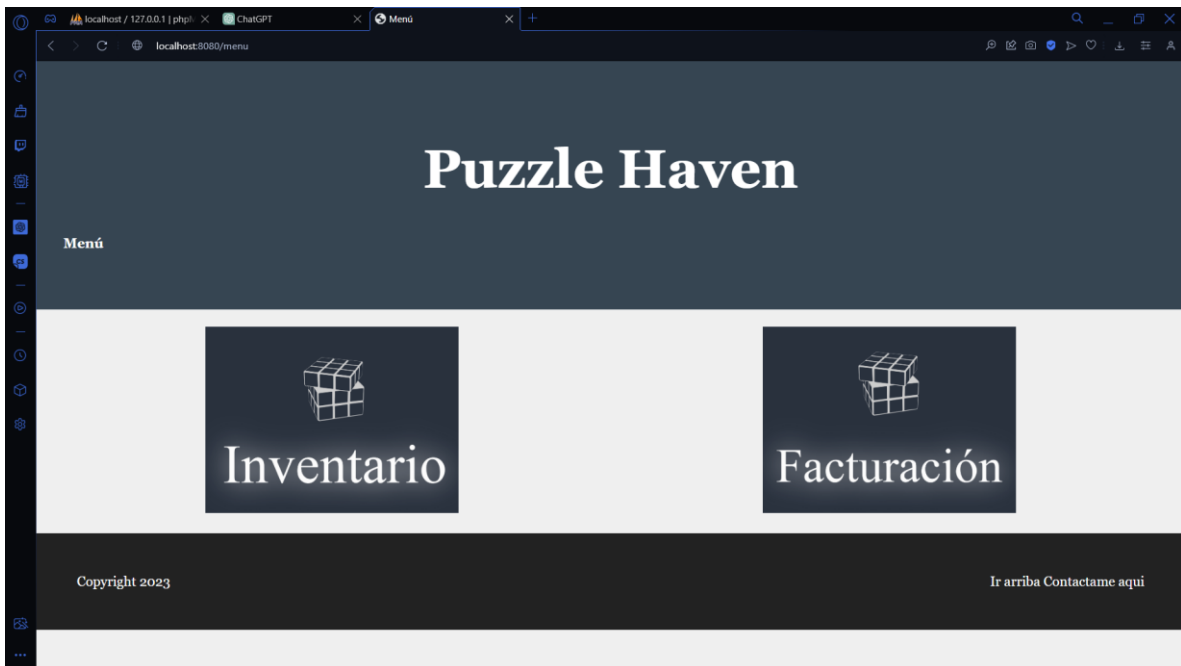


## Estilo

El CSS se utilizó para darle una interfaz agradable al usuario y que sea cómodo usarlo.

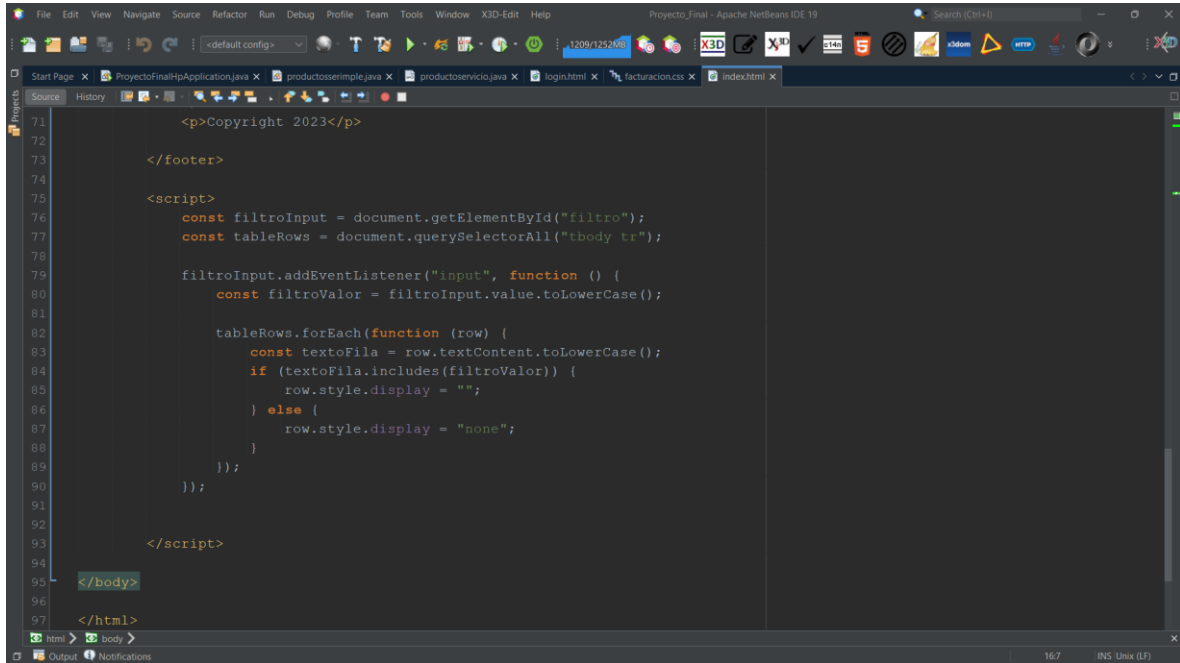


```
11 margin: 0;
12 background-color: #efefef;
13
14 }
15
16 header {
17 background-color: #364552;
18 color: white;
19 padding: 3rem 2rem;
20
21 }
22
23 header h1 {
24 font-size: 4rem;
25 text-align: center;
26
27 }
28
29
30 button.menu {
31 background-color: #34495e; /* Azul oscuro */
32 color: white;
33 padding: 10px 20px;
34 border: none;
35 border-radius: 5px;
36 cursor: pointer;
37 transition: background-color 0.3s ease, transform 0.3s ease;
```



## Funcionamiento

Se utilizó un poco de JavaScript para complementar los funcionamientos de la web.



```
71 <p>Copyright 2023</p>
72
73 </footer>
74
75 <script>
76     const filtroInput = document.getElementById("filtro");
77     const tableRows = document.querySelectorAll("tbody tr");
78
79     filtroInput.addEventListener("input", function () {
80         const filtroValor = filtroInput.value.toLowerCase();
81
82         tableRows.forEach(function (row) {
83             const textoFila = row.textContent.toLowerCase();
84             if (textoFila.includes(filtroValor)) {
85                 row.style.display = "";
86             } else {
87                 row.style.display = "none";
88             }
89         });
90     });
91
92 </script>
93
94 </body>
95
96 </html>
```