



Actividad |3| Funcionamiento del carrito.

Desarrollo de Sistemas Web II.

Ingeniería en Desarrollo de Software.



TUTOR: Aaron Ivan Salazar Macias.

ALUMNO: Ramón Ernesto Valdez Felix.

FECHA: 06/03/2025.

Introducción.....	3
Descripción.....	3
Justificación.....	4
Desarrollo:.....	5
Codificación.....	5
Pruebas del sitio web.....	20
Conexión a la BD.....	23
Conclusion.....	24
Referencias.....	25

Introducción.

En esta actividad final de la materia de Desarrollo de Sistemas Web II, en la evolución constante del proyecto de la tienda Sara, hemos alcanzado un hito significativo: la aprobación del diseño del sitio web. Este logro nos impulsa a la siguiente fase, donde la funcionalidad se convierte en la protagonista. La atención se centra ahora en el corazón del comercio electrónico: el alta de productos y el carrito de compras. Para dotarlo de una operatividad robusta y eficiente, se ha decidido implementar una API REST. Esta interfaz de programación permitirá la manipulación de los datos de productos almacenados en la base de datos, abarcando desde la incorporación de nuevos artículos hasta la modificación, consulta y eliminación de los existentes. La elección de una API REST responde a la necesidad de una comunicación ágil y estructurada entre el frontend del sitio web y el backend, asegurando una experiencia de usuario fluida y sin interrupciones.

Descripción.

En el contexto de la materia Desarrollo de Sistemas Web II, la tienda Sara, en la etapa final del curso el proyecto de la tienda Sara continúa avanzando. Con la aprobación del diseño web, el enfoque se desplaza hacia la funcionalidad, particularmente en el alta de productos y el carrito de compras. Para lograr una gestión eficiente, se implementará una API REST, facilitando la interacción entre el sitio web y la base de datos. Esta API permitirá realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los productos, asegurando una administración completa del catálogo. La elección de una API REST se basa en su capacidad para proporcionar una comunicación ágil y estructurada, esencial para una experiencia de usuario fluida en el entorno dinámico del comercio electrónico. Este enfoque garantiza la escalabilidad y el mantenimiento a largo plazo del sistema. Usamos el contexto de la actividad para la documentación del documento para entregar al curso de esta materia.

Justificación.

En esta actividad trabajaremos con la documentación donde la justificación para el desarrollo del sitio web de la tienda Sara, la decisión de implementar una API REST para gestionar el alta de productos y el carrito de compras de la tienda Sara, se justifica por la necesidad de construir una solución robusta y escalable. En este contexto académico, el uso de una API REST permite aplicar conceptos clave de desarrollo web moderno, como la separación de capas (frontend y backend) y la comunicación basada en estándares (HTTP). Además, esta elección facilita la realización de operaciones CRUD de manera eficiente, optimizando la gestión de la base de datos de productos. La API REST, con su arquitectura flexible, asegura la adaptabilidad del sistema ante futuros requerimientos y la integración con otras funcionalidades. Esta justificación no sólo respalda la implementación técnica, sino que también alinea el proyecto con los objetivos de aprendizaje del curso.

Desarrollo de Sistemas Web II analizando algunos puntos a tomar en cuenta para el llenado de la documentación de esta actividad que son los siguientes:

- PDF de esta actividad en el portafolio GitHub.
- Anexar el archivo comprimido del sitio web en .zip en el portafolio de GitHub.
- Anexa link de GitHub en documento.
- Crear un sistema web tenga las siguientes interfaces.
 - Página de alta de productos.
 - Página consulta del alta de producto.
 - Un menú productos.

Desarrollo:

En este punto realizaremos la documentación de la página de la tienda Sara, incluyendo el diseño, codificación y la conexión a la BD, es fundamental para guiar el desarrollo del sistema y asegurar su correcto funcionamiento. Como información adicional se utilizó una plantilla como referencia y se modificó para llegar al resultado de lo solicitado por la actividad dos de la materia en curso. La actividad se realiza con las instrucciones proporcionadas por el docente de la herramienta. A continuación anexamos los requerimientos proporcionados por la documentación de la actividad.

[Link: GitHub](#)

[Link: SitioWeb](#)

Codificación.

Se agrega la evidencia de la codificación de la creación del sitio web de la tienda Sara, para la actividad final de esta materia de Desarrollo de Sistemas Web II que se está cursando, es fundamental para guiar el desarrollo del sistema y asegurar su correcto funcionamiento.

Codificación Tienda Sara.

Prueba de conexión a Base de Datos de SQL.

En esta pantalla se anexa el submenú del index para ir al sitio crud del alta de los productos.

Controllers.

```
src > main > java > com > tda_sara > Tienda > Sara > Controllers > ProductController.java > Language Support for Java(TM) by Red Hat > ProductController > showProductList(Model)
 1 package com.tda_sara.Tienda.Sara.Controllers;
 2
 3 import java.io.InputStream;
 4 import java.nio.file.Files;
 5 import java.nio.file.Path;
 6 import java.nio.file.Paths;
 7 import java.nio.file.StandardCopyOption;
 8 import java.util.Date;
 9 import java.util.List;
10
11 import org.springframework.beans.factory.annotation.Autowired;
12 import org.springframework.data.domain.Sort;
13 import org.springframework.stereotype.Controller;
14 import org.springframework.ui.Model;
15 import org.springframework.validation.BindingResult;
16 import org.springframework.validation.FieldError;
17 import org.springframework.web.bind.annotation.GetMapping;
18 import org.springframework.web.bind.annotation.ModelAttribute;
19 import org.springframework.web.bind.annotation.PostMapping;
20 import org.springframework.web.bind.annotation.RequestMapping;
21 import org.springframework.web.bind.annotation.RequestParam;
22 import org.springframework.web.multipart.MultipartFile;
23
24 import com.tda_sara.Tienda.Sara.Models.Category;
25 import com.tda_sara.Tienda.Sara.Models.Mark;
26 import com.tda_sara.Tienda.Sara.Models.Product;
27 import com.tda_sara.Tienda.Sara.Models.ProductDto;
28 import com.tda_sara.Tienda.Sara.Services.CategoryRepo;
29 import com.tda_sara.Tienda.Sara.Services.MarkRepo;
30 import com.tda_sara.Tienda.Sara.Services.ProductRepo;
31
32 import jakarta.validation.Valid;
33
34 @Controller
35 @RequestMapping("/products")
36 public class ProductController {
37
38     @Autowired
39     private ProductRepo repo;
40
41     @Autowired
42     private CategoryRepo repo2;
43
44     @Autowired
45     private MarkRepo repo3;
46
47
48
49     @GetMapping({"", "/"})
```

```
src > main > java > com > tda_sara > Tienda > Sara > Controllers > ProductController.java > Language Support for Java(TM) by Red Hat > ProductController > showProductList(Model)
36  public class ProductController {
49      @GetMapping("/list")
50      public String showProductList(Model model) {
51          List<Product> products = repo.findAll(Sort.by(Sort.Direction.DESC, "id"));
52          model.addAttribute("products", products);
53
54
55          return "products/index";
56      }
57
58      public List<Category> getListCategories(){
59          List<Category> list = repo2.findAll(Sort.by(Sort.Direction.DESC, "id"));
60          return list;
61      }
62
63      public List<Mark> getListMarks(){
64          List<Mark> list = repo3.findAll(Sort.by(Sort.Direction.DESC, "id"));
65          return list;
66      }
67
68
69      @GetMapping("/create")
70      public String showCreatePage(Model model) {
71          ProductDto productDto = new ProductDto();
72          model.addAttribute("productDto", productDto);
73          model.addAttribute("categories", getListCategories());
74          model.addAttribute("marks", getListMarks());
75
76          return "products/CreateProduct";
77      }
78
79
80      @PostMapping("/create")
81      public String createProduct(
82          @Valid @ModelAttribute ProductDto productDto,
83          BindingResult result,
84          Model model
85      ) {
86
87          if (productDto.getImageFile().isEmpty()) {
88              result.addError(new FieldError("productDto", "imageFile", "The image file is required"));
89          }
90
91          model.addAttribute("categories", getListCategories());
92          model.addAttribute("marks", getListMarks());
93
94          if (result.hasErrors()) {
95              return "products/CreateProduct";
96          }
97      }
98  }
```

```
src > main > java > com > tda_sara > Tienda > Sara > Controllers > ProductController.java > Language Support for Java(TM) by Red Hat > ProductController > showProductList(Model)
36 <public class ProductController {
37     <public String createProduct(
38
39         // save image file
40         MultipartFile image = productDto.getImageFile();
41         Date createdAt = new Date();
42         String storageFileName = createdAt.getTime() + "_" + image.getOriginalFilename();
43
44         try {
45             String uploadDir = "public/images/";
46             Path uploadPath = Paths.get(uploadDir);
47
48             if (!Files.exists(uploadPath)) {
49                 Files.createDirectories(uploadPath);
50             }
51
52             try (InputStream inputStream = image.getInputStream()) {
53                 Files.copy(inputStream, Paths.get(uploadDir + storageFileName),
54                         StandardCopyOption.REPLACE_EXISTING);
55             }
56         } catch (Exception ex) {
57             System.out.println("Exception: " + ex.getMessage());
58         }
59
60         Product product = new Product();
61         product.setDescription(productDto.getDescription());
62         product.setPrice(productDto.getPrice());
63         product.setAmount(productDto.getAmount());
64         product.setCategory(productDto.getCategory());
65         product.setMark(productDto.getMark());
66         product.setCreatedAt(createdAt);
67         product.setImageFileName(storageFileName);
68
69         repo.save(product);
70
71
72         return "redirect:/products";
73     }
74
75
76     @GetMapping("/edit")
77     public String showEditPage(
78         Model model,
79         @RequestParam int id
80     ) {
81
82         try {
```

```
src > main > java > com > tda_sara > Tienda > Sara > Controllers > ProductController.java > Language Support for Java(TM) by Red Hat > ProductController > showProductList(Model)
 36     public class ProductController {
 37         public String showEditPage(
 38             Model model,
 39             @RequestParam int id) {
 40             Product product = repo.findById(id).get();
 41             model.addAttribute("product", product);
 42             model.addAttribute("categories", getListCategories());
 43             model.addAttribute("marks", getListMarks());
 44
 45             ProductDto productDto = new ProductDto();
 46             productDto.setDescription(product.getDescription());
 47             productDto.setPrice(product.getPrice());
 48             productDto.setAmount(product.getAmount());
 49             productDto.setCategory(product.getCategory());
 50             productDto.setMark(product.getMark());
 51
 52             model.addAttribute("productDto", productDto);
 53
 54         }
 55     }
 56
 57     catch(Exception ex) {
 58         System.out.println("Exception: " + ex.getMessage());
 59         return "redirect:/products";
 60     }
 61
 62     return "products/EditProduct";
 63 }
 64
 65
 66
 67
 68
 69
 70     @PostMapping("/edit")
 71     public String updateProduct(
 72         Model model,
 73         @RequestParam int id,
 74         @Valid @ModelAttribute ProductDto productDto,
 75         BindingResult result
 76     ) {
 77
 78     try {
 79         Product product = repo.findById(id).get();
 80         model.addAttribute("product", product);
 81
 82         if (result.hasErrors()) {
 83             return "products/EditProduct";
 84         }
 85
 86         if (!productDto.getImageFile().isEmpty()) {
 87             // delete old image
 88             String uploadDir = "public/images/";
 89             Path oldImagePath = Paths.get(uploadDir + product.getImageFileName());
 90
 91         }
 92     }
 93
 94     catch (Exception ex) {
 95         System.out.println("Exception: " + ex.getMessage());
 96         return "redirect:/products";
 97     }
 98
 99 }
```

```

src > main > java > com > tda_sara > Tienda > Sara > Controllers > ProductController.java > Language Support for Java(TM) by Red Hat > ProductController > showProductList(Model)
36  public class ProductController {
171  public String updateProduct(
191     try {
192       Files.delete(oldImagePath);
193     }
194     catch(Exception ex) {
195       System.out.println("Exception: " + ex.getMessage());
196     }
197
198     // save new image file
199     MultipartFile image = productDto.getImageFile();
200     Date createdAt = new Date();
201     String storageFileName = createdAt.getTime() + "_" + image.getOriginalFilename();
202
203     try (InputStream inputStream = image.getInputStream()) {
204       Files.copy(inputStream, Paths.get(uploadDir + storageFileName),
205         StandardCopyOption.REPLACE_EXISTING);
206     }
207
208     product.setImageFileName(storageFileName);
209   }
210
211   product.setDescription(productDto.getDescription());
212   product.setPrice(productDto.getPrice());
213   product.setAmount(productDto.getAmount());
214   product.setCategory(productDto.getCategory());
215   product.setMark(productDto.getMark());
216
217   repo.save(product);
218 }
219 catch(Exception ex) {
220   System.out.println("Exception: " + ex.getMessage());
221 }
222
223 return "redirect:/products";
224 }
225
226
227 @GetMapping("/delete")
228 public String deleteProduct(
229   @RequestParam int id
230 ) {
231
232   try {
233     Product product = repo.findById(id).get();
234
235     // delete product image
236     Path imagePath = Paths.get("public/images/" + product.getImageFileName());
237

```

```

src > main > java > com > tda_sara > Tienda > Sara > Controllers > ProductController.java > Language Support for Java(TM) by Red Hat > ProductController > showProductList(Model)
36  public class ProductController {
228  public String deleteProduct(
238
239    try {
240      Files.delete(imagePath);
241    }
242    catch(Exception ex) {
243      System.out.println("Exception: " + ex.getMessage());
244    }
245
246
247    // delete the product
248    repo.delete(product);
249  }
250  catch (Exception ex) {
251    System.out.println("Exception: " + ex.getMessage());
252  }
253
254  return "redirect:/products";
255 }
256

```

Models.

```
src > main > java > com > tda_sara > Tienda > Sara > Models > Category.java > {} com.tda_sara.Tienda.Sara.Models
 1 package com.tda_sara.Tienda.Sara.Models;
 2
 3 import jakarta.persistence.Column;
 4 import jakarta.persistence.Entity;
 5 import jakarta.persistence.GeneratedValue;
 6 import jakarta.persistence.GenerationType;
 7 import jakarta.persistence.Id;
 8 import jakarta.persistence.Table;
 9
10 @Entity
11 @Table(name="Categorias")
12 public class Category {
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private int id;
16
17     @Column(columnDefinition = "TEXT")
18     private String description;
19
20     public int getId() {
21         return id;
22     }
23
24     public void setId(int id) {
25         this.id = id;
26     }
27
28     public String getDescription() {
29         return description;
30     }
31
32     public void setDescription(String description) {
33         this.description = description;
34     }
35
36
37
38 }
```

```
src > main > java > com > tda_sara > Tienda > Sara > Models > CategoryDto.java > CategoryDto
 1 package com.tda_sara.Tienda.Sara.Models;
 2
 3 import jakarta.validation.constraints.*;
 4
 5 public class CategoryDto {
 6     @Size(min = 10, message = "The description should be at least 10 characters")
 7     @Size(max = 2000, message = "The description cannot 2000 characters")
 8     private String description;
 9
10     public String getDescription() {
11         return description;
12     }
13
14     public void setDescription(String description) {
15         this.description = description;
16     }
17 }
```

```
src > main > java > com > tda_sara > Tienda > Sara > Models > J Mark.java > Mark
 1 package com.tda_sara.Tienda.Sara.Models;
 2
 3 √ import jakarta.persistence.Column;
 4 import jakarta.persistence.Entity;
 5 import jakarta.persistence.GeneratedValue;
 6 import jakarta.persistence.GenerationType;
 7 import jakarta.persistence.Id;
 8 import jakarta.persistence.Table;
 9
10 @Entity
11 @Table(name="Marcas")
12 √ public class Mark {
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private int id;
16
17     @Column(columnDefinition = "TEXT")
18     private String description;
19
20 √     public int getId() {
21         return id;
22     }
23
24 √     public void setId(int id) {
25         this.id = id;
26     }
27
28 √     public String getDescription() {
29         return description;
30     }
31
32 √     public void setDescription(String description) {
33         this.description = description;
34     }
35 }
```

```
src > main > java > com > tda_sara > Tienda > Sara > Models > J MarkDto.java > MarkDto
1  package com.tda_sara.Tienda.Sara.Models;
2
3  import jakarta.validation.constraints.*;
4
5  public class MarkDto {
6      @Size(min = 10, message = "The description should be at least 10 characters")
7      @Size(max = 2000, message = "The description cannot 2000 characters")
8      private String description;
9
10     public String getDescription() {
11         return description;
12     }
13
14     public void setDescription(String description) {
15         this.description = description;
16     }
17 }
```

```
src > main > java > com > tda_sara > Tienda > Sara > Models > Product.java > ...
1  package com.tda_sara.Tienda.Sara.Models;
2
3  import java.util.Date;
4
5  import jakarta.persistence.Column;
6  import jakarta.persistence.Entity;
7  import jakarta.persistence.GeneratedValue;
8  import jakarta.persistence.GenerationType;
9  import jakarta.persistence.Id;
10 import jakarta.persistence.JoinColumn;
11 import jakarta.persistence.ManyToOne;
12 import jakarta.persistence.Table;
13
14 @Entity
15 @Table(name="Productos")
16 public class Product {
17     @Id
18     @GeneratedValue(strategy = GenerationType.IDENTITY)
19     private int id;
20
21     @Column(columnDefinition = "TEXT")
22     private String description;
23
24     @Column(name="Precio")
25     private double price;
26
27     @Column(name="Cantidad")
28     private int amount;
29
30     @ManyToOne()
31     @JoinColumn(name="IdCategoria")
32     private Category category;
33
34     @ManyToOne()
35     @JoinColumn(name="IdMarca")
36     private Mark mark;
37
38     private Date createdAt;
39     private String imageFileName;
40
41     public int getId() {
42         return id;
43     }
44     public void setId(int id) {
45         this.id = id;
46     }
47     public String getDescription() {
48         return description;
49     }
```

```
src > main > java > com > tda_sara > Tienda > Sara > Models > Product.java > ...
16  public class Product {
49
50      ...
51      public void setDescription(String description) {
52          this.description = description;
53      }
54      public double getPrice() {
55          return price;
56      }
57      public void setPrice(double price) {
58          this.price = price;
59      }
60      public int getAmount() {
61          return amount;
62      }
63      public void setAmount(int amount) {
64          this.amount = amount;
65      }
66      public Category getCategory() {
67          return category;
68      }
69      public void setCategory(Category category) {
70          this.category = category;
71      }
72      public Mark getMark() {
73          return mark;
74      }
75      public void setMark(Mark mark) {
76          this.mark = mark;
77      }
78      public Date getCreatedAt() {
79          return createdAt;
80      }
81      public void setCreatedAt(Date createdAt) {
82          this.createdAt = createdAt;
83      }
84      public String getImageFileName() {
85          return imageName;
86      }
87      public void setImageFileName(String imageName) {
88          this.imageName = imageName;
89  }
```

```
src > main > java > com > tda_sara > Tienda > Sara > Models > ProductDto.java > ...
1  package com.tda_sara.Tienda.Sara.Models;
2
3  import org.springframework.web.multipart.MultipartFile;
4
5  import jakarta.validation.constraints.*;
6
7  public class ProductDto {
8      @Size(min = 10, message = "The description should be at least 10 characters")
9      @Size(max = 2000, message = "The description cannot 2000 characters")
10     private String description;
11
12     @Min(0)
13     private double price;
14
15     @Min(0)
16     private int amount;
17
18     private Category category;
19
20     private Mark mark;
21
22     private MultipartFile imageFile;
23
24     public String getDescription() {
25         return description;
26     }
27
28     public void setDescription(String description) {
29         this.description = description;
30     }
31
32     public double getPrice() {
33         return price;
34     }
35
36     public void setPrice(double price) {
37         this.price = price;
38     }
39
40     public int getAmount() {
41         return amount;
42     }
43
44     public void setAmount(int amount) {
45         this.amount = amount;
46     }
47
48     public Category getCategory() {
49         return category;
```

```

src > main > java > com > tda_sara > Tienda > Sara > Models > J ProductDto.java > ↗ ProductDto > ↗ imageFile
 7   public class ProductDto {
48     public Category getCategory() {
50       }
51
52       public void setCategory(Category category) {
53         this.category = category;
54       }
55
56       public Mark getMark() {
57         return mark;
58       }
59
60       public void setMark(Mark mark) {
61         this.mark = mark;
62       }
63
64       public MultipartFile getImageFile() {
65         return imageFile;
66       }
67
68       public void setImageFile(MultipartFile imageFile) {
69         this.imageFile = imageFile;
70       }
71     }
72

```

Services.

```

src > main > java > com > tda_sara > Tienda > Sara > Services > J CategoryRepo.java > Language Support for Java(TM) by Red Hat > ↗ CategoryRepo
 1 package com.tda_sara.Tienda.Sara.Services;
 2
 3 import org.springframework.data.jpa.repository.JpaRepository;
 4 import com.tda_sara.Tienda.Sara.Models.Category;
 5
 6 public interface CategoryRepo extends JpaRepository<Category, Integer>{
 7
 8 }

```

```

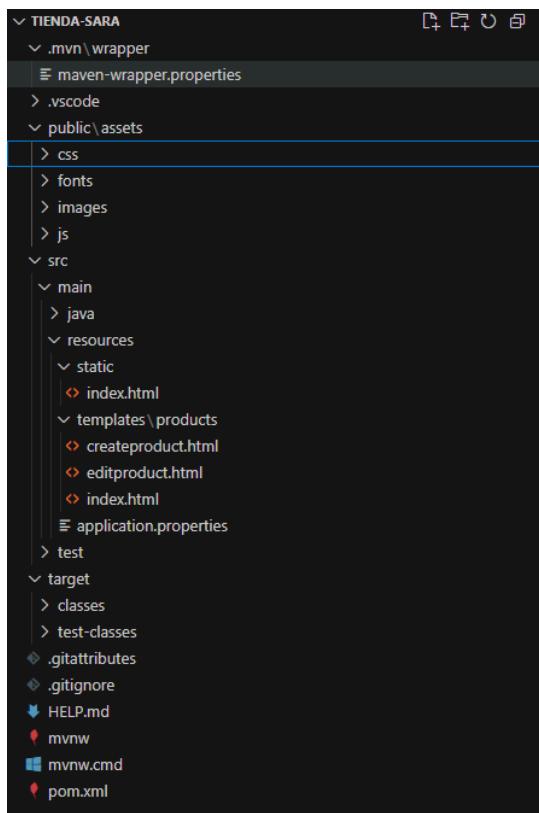
src > main > java > com > tda_sara > Tienda > Sara > Services > J MarkRepo.java > Spring Boot Tools > ↗ @+ 'markRepo' (Mark) Repository<Mark, Integer>
 1 package com.tda_sara.Tienda.Sara.Services;
 2
 3 import org.springframework.data.jpa.repository.JpaRepository;
 4 import com.tda_sara.Tienda.Sara.Models.Mark;
 5
 6 public interface MarkRepo extends JpaRepository<Mark, Integer>{
 7
 8 }

```

```

src > main > java > com > tda_sara > Tienda > Sara > Services > J ProductRepo.java > Spring Boot Tools > ↗ @+ 'productRepo' (Product) Repository<Product, Integer>
 1 package com.tda_sara.Tienda.Sara.Services;
 2
 3 import org.springframework.data.jpa.repository.JpaRepository;
 4 import com.tda_sara.Tienda.Sara.Models.Product;
 5
 6 public interface ProductRepo extends JpaRepository<Product, Integer> {
 7
 8 }

```



Pruebas del sitio web.

Se anexará la evidencia del sitio web de la tienda Sara, en un resumen general el sitio principal nos lleva a los productos donde se dará de alta el producto de la tienda sara.

Sitio web: “Tienda SARA”.

Página Index.html: a la cual se le agregó un submenú de nombre Productos.

Alta de los productos en CRUD del sitio de la tienda Sara.

Tienda SARA

localhost:8081/products/create

New Product

Description	Laptop Gamer Lenovo LOQ 15ARP9 15.6 Pulgadas Windows 11 AMD Ryzen 5 NVIDIA GeForce RTX 3050 8 GB RAM 512 GB SSD Gr.
Price	20000
Amount	1
Category	Computo
Mark	Lenovo
Image	Seleccionar archivo computo1.jpg
<input type="button" value="Submit"/> <input type="button" value="Cancel"/>	

Se muestran los productos dados de alta CRUD del sitio de la tienda Sara.

Tienda SARA

localhost:8081/products

Products

Create Product

ID	Description	Price	Amount	Category	Mark	Image	Created At	Action
8	Zapatillas clásicas con estilo retro y comodidad excepcional.	\$2800.0	1	Zapatos	Nike		2025-03-09	Edit Delete
7	Ropa deportiva cómoda y duradera para tus entrenamientos.	\$1900.0	1	Ropa	Nike		2025-03-09	Edit Delete
6	Smartphone con cámara de alta resolución y rendimiento potente.	\$28999.0	1	Moviles	Samsung		2025-03-09	Edit Delete
5	Smart TV con pantalla QLED y sonido envolvente para una experiencia cinematográfica.	\$10000.0	1	Electronica	Samsung		2025-03-09	Edit Delete
4	Smart TV con resolución 4K y tecnología HDR para imágenes vibrantes	\$11000.0	1	Electronica	Hisense		2025-03-09	Edit Delete
3	ASUS Laptop ROG Strix G14/16 FHD+/Intel Core i7 13va/NVIDIA GeForce RTX4060/16GB RAM Expandible hasta 32GB/1TB SSD/Teclado Retroiluminado RGB 4	\$25000.0	1	Computo	ASUS		2025-03-09	Edit Delete
2	Laptop Dell Gaming G15 5530 Ci5-13450HX 10 núcleos 16 GB 512 GB SSD RTX4050.	\$24000.0	1	Computo	DELL		2025-03-09	Edit Delete
1	Laptop Gamer Lenovo LOQ 15ARP9 15.6 Pulgadas Windows 11 AMD Ryzen 5 NVIDIA GeForce RTX 3050 8 GB RAM 512 GB SSD Gr.	\$20000.0	1	Computo	Lenovo		2025-03-09	Edit Delete

Conexión a la BD.

La actividad de conexión de las bases de datos con la aplicación web y la herramienta de programación de visual code, con esto se da la finalización del requerimiento del documento de creación de sitio web de la tienda sara

Conexiona la base de datos del sitio de la tienda Sara.

The screenshot shows the SSMS interface. In the Object Explorer, the database 'SARAa3' is selected, displaying its structure with tables like 'dbo.categorias', 'dbo.marcas', and 'dbo.productos'. In the bottom pane, a query window runs a SELECT statement on the 'categorias' table, returning five rows of data:

	id	description
1	1	Computo
2	2	Electronica
3	3	Moviles
4	4	Ropa
5	5	Zapatos

The status bar at the bottom indicates the query was executed successfully.

The screenshot shows two separate SQL queries run in SQL Server Management Studio.

Query 1:

```
SQLQuery2.sql - DE...SS.SARAa3 (sa (60)) [SELECT TOP (1000) [id], [description] FROM [SARAa3].[dbo].[marcas]]
```

Results:

	id	description
1	1	Lenovo
2	2	DELL
3	3	ASUS
4	4	Hisense
5	5	Samsung
6	6	LG
7	7	Honor
8	8	iPhone
9	9	Nike

Query 2:

```
SQLQuery3.sql - DE...SS.SARAa3 (sa (56)) [SELECT TOP (1000) [id], [cantidad], [created_at], [description], [image_file_name], [precio], [id_categoria], [id_marca] FROM [SARAa3].[dbo].[productos]]
```

Results:

	id	cantidad	created_at	description	image_file_name	precio	id_categoria	id_marca
1	1	1	2025-03-09 23:30:43.135000	Laptop Gamer Lenovo Legion 15ARPP 15.6 Pulgadas W...	1741584643135_computo1.jpg	20000	1	1
2	2	1	2025-03-09 23:32:43.931000	Laptop Dell Gaming G15 5530 Ci5-13450HX 10 núcleos...	1741584763331_computo2.jpg	24000	1	2
3	3	1	2025-03-09 23:33:38.044000	ASUS Laptop ROG Strix G14/16 FHD+ Intel Core i7 13v...	1741584818044_computo3.jpg	25000	1	3
4	4	1	2025-03-09 23:34:52.176000	Smart TV con resolución 4K y tecnología HDR para imág...	17415848932176_electronica1.jpg	11000	2	4
5	5	1	2025-03-09 23:35:41.541000	Smart TV con pantalla QLED y sonido envolvente para u...	1741584941541_electronica2.jpg	10000	2	5
6	6	1	2025-03-09 23:36:52.655000	Smartphone con cámara de alta resolución y rendimiento...	1741585012655_movies1.jpg	2899	3	5
7	7	1	2025-03-09 23:37:50.143000	Ropa deportiva cómoda y duradera para tus entrenamientos...	1741585070143_ropa1.jpg	1900	4	9
8	8	1	2025-03-09 23:38:31.375000	Zapatos clásicos con estilo retro y comodidad excepcional...	1741585111375_zapatos1.jpg	2800	5	9

Conclusion.

En conclusión: En esta actividad se presentaron muchos errores al tratar de realizar las configuraciones del sitio de sara, errores de conexión con la base de datos ya que no se contaba con el puerto de sql 1433 no activado, la conexión con autenticación con sql, en visual code también presentó errores con los componentes requeridos para poder trabajar con la actividad como la herramienta de openjdk 17 que no se instalaba en automático, error de levantamiento del puerto del sitio web. Con todo estos errores presentados en la actividad nos enseña a extraer preparados de manera correcta para realizar las actividades de manera más rápida y eficaz. En resumen: Un buen diseño web y la conexión

a Internet son esenciales para el éxito laboral y personal. Facilitan la comunicación, la productividad y el acceso a la información, conectándonos con el mundo y permitiendo el aprendizaje, la socialización y el entretenimiento. En una era digital, estas herramientas son clave para el desarrollo personal y profesional.

Referencias.

Gemini - chat to supercharge your ideas. (n.d.). Gemini. Retrieved January 9, 2025,

from <https://gemini.google.com/>

Ingeniería en desarrollo de software. (n.d.). Edu.Mx. Retrieved January 9, 2025, from

<https://umi.edu.mx/coppel/IDS/login/index.php>