Informe de Laboratorio 09

Tema: Proyecto Final: Aplicación web para una tienda de dispositivos electronicos

Nota

Estudiantes	Escuela	Asignatura
- Alejandro Sebastian Alfonso Huacasi aalfonso@unsa.edu.pe - Mauricio Elias Congona Manrique mcongonam@unsa.edu.pe - Klismann Chancuaña Alvis kchancuana@unsa.edu.pe - Maxs Sebastian Joaquin Forocca Mamani mforocca@unsa.edu.pe	Escuela Profesional de Ingenieria de Sistemas	Programación web 2 Semestre: A

Laboratorio	Tema	Duración
09	Proyecto Final	04 horas

Semestre académico:	Fecha de inicio	Fecha de entrega
2023 - A	Del 25 Julio 2023	Al 05 Agosto 2023

TAREA

- Integrar los conocimientos adquiridos en el curso
- Trabajar en equipo
- Desarrollar una aplicación Web para una empresa real, que incluya: Backend (Django) y FrontEnd (Ajax / Framework JavaScript: Angular)

1. Equipos, materiales y temas utilizados

- Sistema Operativo (GNU/Linux de preferencia).
- GNU Vim.
- Python 3
- Git.
- Cuenta en GitHub con el correo institucional.

- Ajax
- Django 4
- JavaScript
- Angular

2. URL DEL REPOSITORIO GRUPAL DE GIT HUB Y LINK DEL VIDEO

- `https://github.com/MauCausa/Pweb2-Lab-B.git`
- `https://drive.google.com/file/d/1q9kaKKUeCIjJ-7oXO2CRWyTsanYFj_ZI/view?usp=sharing`

3. Actividades con el repositorio GitHub

3.1. Creando e inicializando repositorio GitHub

- Como ya tenemos nuestro repositorio GitHub y además esta clonado e inicializado en nuestra máquina local.
- Se realizaron los siguientes comandos en la computadora:

Listing 1: Creando carpeta de trabajo dentro de nuestro repositorio clonado en mi maquina local

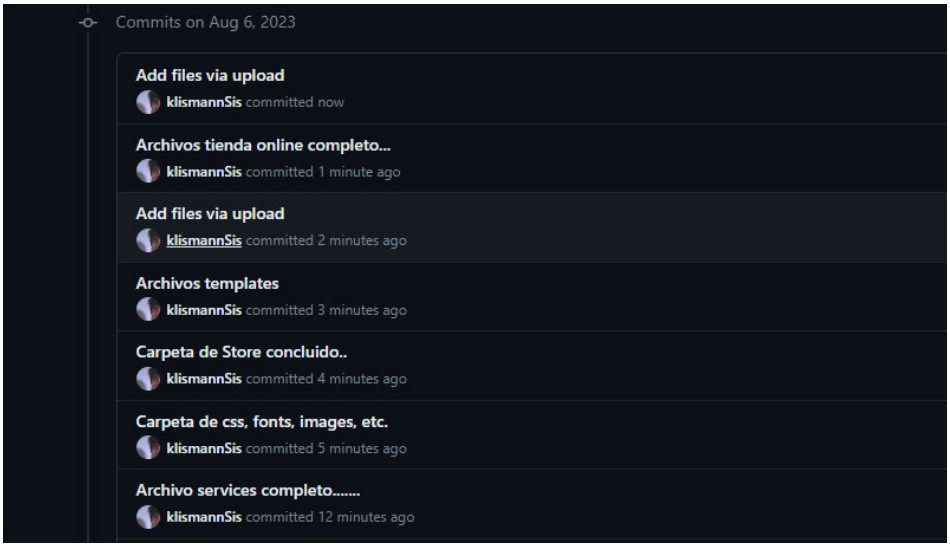
```
$ cd Pweb2-Lab-B
$ mkdir ProyectoFinal
```

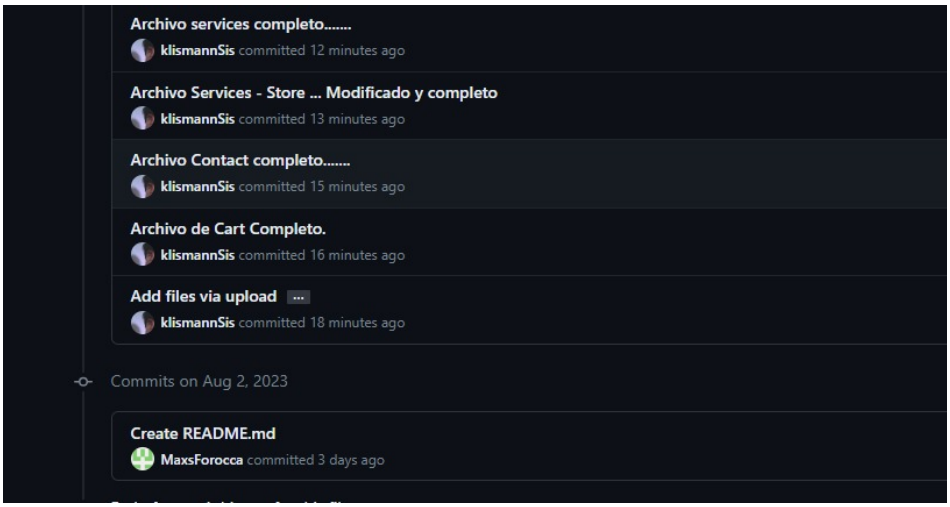
Listing 2: Dirijiéndonos a la carpeta de trabajo

```
$ cd ProyectoFinal
```

3.2. Commits

- A continucación se mostraran capturas de los commits:





- Explicaremos algunos archivos de nuestro proyecto final:

Listing 3: apps.py, se encuentra en la carpeta tiendaOnline, dentro de la carpeta de cart

```
1 from django.apps import AppConfig
2
3
4 class CartConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'cart'
```

- from django.apps import AppConfig: Esta línea importa la clase AppConfig del módulo django.apps. AppConfig es una clase de Django que se utiliza para configurar y personalizar la aplicación.
- class CartConfig(AppConfig):: Aquí defines una nueva clase llamada CartConfig que hereda de

AppConfig. Esto significa que estás creando una configuración personalizada para una parte específica de tu aplicación.

- defaultautofield = 'django.db.models.BigAutoField': Estableces el campo automático predeterminado que se utilizará para las claves primarias en los modelos de esta aplicación. En este caso, estás utilizando un campo llamado BigAutoField, que es una variante de clave primaria automática que admite valores grandes.
- name = 'cart': Aquí defines el nombre de la aplicación como 'cart'. Esto es cómo Django identificará y referenciará la aplicación en varias partes de la configuración y el código.

Listing 4: cart.py, se encuentra en la carpeta tiendaOnline, dentro de la carpeta de cart

```
class Cart:
    def __init__(self, request):
        self.request = request
        self.session = request.session
        carro = self.session.get("carro")
        if not carro:
            carro = self.session["carro"]={}

        self.carro = carro

    def agregar(self,producto):
        if(str(producto.id) not in self.carro.keys()):
            self.carro[producto.id]={
                "producto_id":producto.id,
                "nombre":producto.nombre,
                "precio":str(producto.precio),
                "cantidad":1,
                "imagen":producto.imagen.url
            }
        else:
            for key, value in self.carro.items():
                if key==str(producto.id):
                    value["cantidad"]=value["cantidad"]+1
                    value["precio"]=float(value["precio"])+producto.precio
                    break
        self.guardar_carro()

    def guardar_carro(self):
        self.session["carro"]=self.carro
        self.session.modified=True

    def eliminar(self,producto):
        producto.id=str(producto.id)
        if producto.id in self.carro:
            del self.carro[producto.id]
            self.guardar_carro()

    def restar_producto(self, producto):
        for key, value in self.carro.items():
                if key==str(producto.id):
                    value["precio"]=float(value["precio"])-producto.precio
                    value["cantidad"]=value["cantidad"]-1
                    if value["cantidad"]<1:
```

```
                    self.eliminar(producto)
                break
        self.guardar_carro()

    def limpiar_carro(self):
        self.session["carro"]={}
        self.session.modified=True
```

- init (self, request): El constructor de la clase Cart toma un objeto request como argumento. Inicializa el carrito de compras utilizando la sesión del usuario. Si no hay un carrito en la sesión, se crea uno vacío.
- agregar(self, producto): Agrega un producto al carrito. Si el producto no está en el carrito, se crea una entrada en el diccionario del carrito con detalles como el ID del producto, el nombre, el precio, la cantidad y la imagen. Si el producto ya está en el carrito, se incrementa la cantidad y se ajusta el precio total.
- guardarcarro(self): Actualiza la sesión del usuario con los contenidos actuales del carrito. Se utiliza para guardar los cambios realizados en el carrito.
- eliminar(self, producto): Elimina un producto específico del carrito si está presente.
- restarproducto(self, producto): Reduce la cantidad de un producto en el carrito. Si la cantidad se reduce a cero o menos, el producto se elimina del carrito.
- limpiarcarro(self): Vacía completamente el carrito, eliminando todos los productos de la sesión.

Listing 5: urls.py, se encuentra en la carpeta tiendaOnline, dentro de cart

```
from django.urls import path
from .import views

app_name="carro"

urlpatterns = [
    path("add/<int:product_id>/", views.add_product, name="add"),
    path("delete/<int:product_id>/", views.delete_product, name="delete"),
    path("subtract/<int:product_id>/", views.subtract_product, name="subtract"),
    path("clear/", views.clear_cart, name="clear"),
]
```

- from django.urls import path: Esto importa la función path de django.urls, que se utiliza para definir rutas en la aplicación web.
- from .import views: Importa las vistas (funciones) relacionadas con el carrito de compras desde el mismo directorio en el que se encuentra este archivo.
- appname=çarro": Define un espacio de nombres (namespace) llamado çarro"para estas URLs. Esto permite diferenciarlas de las URLs de otras partes de la aplicación.
- urlpatterns: Aquí se definen las rutas URL junto con las vistas que deben ejecutarse cuando se accede a esas rutas.

Listing 6: views.py, se encuentra en la carpeta tiendaOnline, dentro de la carpeta de cart

```python
from django.shortcuts import render
from .cart import Cart
from store.models import Product
from django.shortcuts import redirect

def add_product(request, product_id):

    carro = Cart(request)
    product = Product.objects.get(id=product_id)
    carro.agregar(product=product)

    return redirect("store")


def delete_product(request, product_id):
    carro = Cart(request)
    product = Product.objects.get(id=product_id)
    carro.eliminar(product=product)

    return redirect("store")


def subtract_product(request, product_id):
    carro = Cart(request)
    product = Product.objects.get(id=product_id)
    carro.restar_producto(product=product)

    return redirect("store")


def clear_cart(request, product_id):
    carro = Cart(request)
    carro.limpiar_carro()

    return redirect("store")
```

- add product(request, product id): Esta vista se ejecuta cuando un usuario agrega un producto al carrito. Se instancia un objeto Cart utilizando la solicitud (request). Luego, se recupera el producto correspondiente al ID proporcionado desde la base de datos. A continuación, se llama al método agregar del carrito para agregar el producto. Finalmente, la vista redirige al usuario a una página o ruta llamada "store".
- delete product(request, product id): Esta vista se ejecuta cuando un usuario elimina un producto del carrito. Igual que antes, se instancia un objeto Cart y se obtiene el producto desde la base de datos. Se llama al método eliminar del carrito para quitar el producto, y luego se redirige al usuario a "store".
- subtract product(request, product id): Esta vista se activa cuando un usuario resta la cantidad de un producto en el carrito. Se crea un objeto Cart, se recupera el producto y se llama al método restar producto para reducir la cantidad. Después, se redirige al usuario a "store".
- clear cart(request, product id): Esta vista se ejecuta cuando un usuario desea vaciar completamente el carrito. Se crea un objeto Cart y se llama al método limpiar carro para eliminar todos los productos. Luego, el usuario es redirigido a "store".

Listing 7: contextsprocessor.py, se encuentra en la carpeta tiendaOnline, dentro de la carpeta cart

```python
def importe_total_carro(request):
    total = 0
    if request.user.is_authenticated:
        for key, value in request.session["carro"].items():
            total=total+float(value["precio"])
    else:
        total="Debes hacer login"

    return {"importe_total_carro":total}
```

- La función importe total carro(request) toma una solicitud (request) como argumento.
- La variable total se inicializa en 0. Esta variable se utilizará para calcular el importe total del carrito.
- Se verifica si el usuario está autenticado (logueado) utilizando request.user.is authenticated. Si el usuario está autenticado, el código dentro de este bloque se ejecutará.
- Un bucle for itera a través de los elementos en request.session[çarro"]. Esto asume que el carrito está almacenado en la sesión del usuario bajo la clave çarro".
- En cada iteración, el precio del producto se extrae de value["precio"] y se convierte a un valor numérico usando float(). Luego, este valor se suma al total acumulado.
- Si el usuario no está autenticado, se establece total en un mensaje de "Debes hacer login".
- Al final, la función devuelve un diccionario con una sola entrada ïmporte total carro"que contiene el total calculado o el mensaje de inicio de sesión requerido, dependiendo del estado de autenticación del usuario.

Listing 8: forms.py, se encuentra en la carpeta tiendaOnline, dentro de la carpeta Contact

```python
from django import forms

class ContactForm(forms.Form):
    name = forms.CharField(label='',required=True, widget=forms.TextInput(
        attrs = {
            'placeholder' : 'Your name',
            'id' : 'name',
        }
    ))
    email = forms.CharField(label="", required=True, widget=forms.TextInput(
        attrs = {
            'placeholder' : 'Your email',
            'id' : 'email',
        }
    ))
    message = forms.CharField(label="", widget=forms.Textarea(
        attrs = {
            'placeholder' : 'Your message',
            'id' : 'message',
            'class' : 'col-lg-12',
        }
    ))
```

- La clase ContactForm hereda de forms.Form, que es una clase base para la creación de formularios en Django.
- Se definen tres campos del formulario: name, email y message.
- name y email son campos de entrada de texto (CharField). Se les proporciona un atributo widget personalizado (forms.TextInput) que agrega atributos HTML, como el atributo de marcador de posición (placeholder) y el ID del elemento (id).
- message es un campo de área de texto (CharField). Se utiliza un widget forms.Textarea que permite la entrada de texto más larga. Al igual que en los campos anteriores, se definen atributos como el marcador de posición, el ID y una clase CSS para el estilo.
- Los atributos label en los campos se establecen en una cadena vacía o en blanco (label=''), lo que indica que no se mostrarán etiquetas por defecto en el formulario. Esto es común cuando se utilizan marcadores de posición dentro de los campos como indicadores visuales para el usuario.

Listing 9: admin.py, se encuentra en la carpeta tiendaOnline, dentro de la carpeta Services

```python
from django.contrib import admin
from .models import Service

class ServicesAdmin(admin.ModelAdmin):
    readonly_fields = ('created', 'updated')

admin.site.register(Service, ServicesAdmin)
```

- from django.contrib import admin: Esto importa el módulo de administración de Django, que proporciona la interfaz de administración para gestionar los modelos y los datos de la base de datos.
- from .models import Service: Importa el modelo Service desde el mismo directorio donde se encuentra este archivo.
- class ServicesAdmin(admin.ModelAdmin): Define una clase ServicesAdmin que hereda de admin.ModelAdmin. Esta clase personaliza la forma en que se muestra el modelo Service en el panel de administración.
- readonly fields = ('created', 'updated'): Esta línea establece los campos que se mostrarán en modo de solo lectura en la interfaz de administración. Los campos created y updated parecen ser campos de fecha y hora que indican cuándo se creó y actualizó un servicio.
- admin.site.register(Service, ServicesAdmin): Aquí se registra el modelo Service en el panel de administración de Django. Se utiliza la clase ServicesAdmin personalizada para configurar cómo se mostrará y se administrará el modelo.

Listing 10: models.py, se encuentra en la carpeta tiendaOnline, dentro de la carpeta Store

```python
from django.db import models

class Category(models.Model):
    name = models.CharField(max_length=50)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now_add=True)

```

```
    def __str__(self):
        return self.name


class Product(models.Model):
    name = models.CharField(max_length=50)
    categories = models.ForeignKey(Category, on_delete=models.CASCADE)
    image = models.ImageField(upload_to="store", null=True, blank=True)
    price = models.FloatField()
    stock = models.IntegerField(default=True)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.name
```

- Category es un modelo que representa una categoría de productos. Tiene un campo de texto name que almacena el nombre de la categoría. También tiene campos created y updated para registrar cuándo se creó y actualizó la categoría. El método str devuelve el nombre de la categoría para su representación en la interfaz de administración.
- Product es un modelo que representa un producto. Tiene un campo de texto name para almacenar el nombre del producto. El campo categories es una clave externa (ForeignKey) que enlaza el producto con una categoría. El campo image es una imagen del producto, almacenada en la carpeta "store". price almacena el precio del producto como un número decimal de punto flotante. stock almacena la cantidad disponible en stock. Los campos created y updated registran las fechas de creación y actualización. El método str devuelve el nombre del producto.

Listing 11: login, se encuentra en la carpeta templates, dentro de la carpeta accounts

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>Login</title>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f2f2f2;
            background-image: url("/static/img/four.jpg");
            background-size: cover;
            margin: 0;
            padding: 0;
        }

        form {
            max-width: 300px;
            margin: 20px auto;
            padding: 30px;
            background-color: rgba(300, 300, 300, 0.20); /* Color de fondo transparente */
            border-radius: 5px;
            box-shadow: 0 0 5px rgba(0, 0, 0, 0.3);
            color: #fff;
```

```
        }

        input {
            display: block;
            width: 95%;
            padding: 10px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 3px;
            background-color: rgba(500, 500, 500, 0.10); /* Color de fondo transparente */
            border-radius: 5px;
            box-shadow: 0 0 5px rgba(0, 0, 0, 0.3);
            color: white;
            transition: box-shadow 0.10s;
        }

        input[type="submit"] {
            background-color: #008080;
            color: #fff;
            border: none;
            cursor: pointer;
        }

        input:hover {
            box-shadow: 0 0 20px #4CAF50;
        }

        div {
            max-width: 300px;
            margin: 10px auto;
            padding: 20px;
            background-color: #f2f2f2;
            border: 1px solid #ccc;
            border-radius: 5px;
        }

        h3 {
            color: #4CAF50;
            margin: 5px;
        }
    </style>
</head>

<body>
    <form action="login" method="POST">
        {% csrf_token %}
        <input type="text" name="username" placeholder="username"><br>
        <input type="password" name="password" placeholder="password"><br>
        <input type="Submit">
    </form>

    <div>
        {% for message in messages %}
            <h3>{{ message }}</h3>
        {% endfor %}
    </div>
```

```
</body>

</html>
```

- ¡form action=”login”method=”POST»: El formulario envía los datos del usuario a la URL ”loginütilizando el método POST. Esto se espera que esté configurado en tus URLs de Django.
- ¡input type=”text”name=üsername”placeholder=üsername»¡br¿: Un campo de entrada de texto para el nombre de usuario. Los atributos name y placeholder están configurados.
- ¡input type=”password”name=”password”placeholder=”password»¡br¿: Un campo de entrada de texto para la contraseña, con el atributo type configurado como ”password”.
- ¡input type=”Submit»: Un botón de envío del formulario.

Listing 12: register, se encuentra en la carpeta templates, dentro de la carpeta accounts

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <style>


        body {
            font-family: Arial, sans-serif;
            background-color: #f2f2f2;
            background-image: url("/static/img/two.jpg");
            background-size: cover;
            margin: 0;
            padding: 0;


        }

        form {
            max-width: 300px;
            margin: 20px auto;
            padding: 30px;
            background-color: rgba(300, 300, 300, 0.30); /* Color de fondo transparente */
            border-radius: 5px;
            box-shadow: 0 0 5px rgba(0, 0, 0, 0.3);
            color: #fff;

        }

        input {
            display: block;
            width: 95%;
            padding: 10px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 3px;
            background-color: rgba(500, 500, 500, 0.10); /* Color de fondo transparente */
```

```
            border-radius: 5px;
            box-shadow: 0 0 5px rgba(0, 0, 0, 0.3);
            color: white;
            transition: box-shadow 0.10s;

        }

        input[type="submit"] {
            background-color: #008080;
            color: #fff;
            border: none;
            cursor: pointer;
        }
        input:hover{
            box-shadow: 0 0 20px #4CAF50;
        }

        div {
            max-width: 300px;
            margin: 10px auto;
            padding: 20px;
            background-color: #f2f2f2;
            border: 1px solid #ccc;
            border-radius: 5px;
        }

        h3 {
            color: #4CAF50;
            margin: 5px;
        }
    </style>
</head>
<body>
    <form action="register" method="POST"> {% csrf_token %}
        <input type="text" name="first_name" placeholder="First Name"><br>
        <input type="text" name="last_name" placeholder="Last Name"><br>
        <input type="text" name="username" placeholder="Username"><br>
        <input type="email" name="email" placeholder="Email"><br>
        <input type="password" name="password1" placeholder="Password"><br>
        <input type="password" name="password2" placeholder="Confirm Password"><br>
        <input type="Submit">
    </form>

    <div>
        {% for message in messages %}
            <h3>{{ message }}</h3>
        {% endfor %}
    </div>
</body>
</html>
```

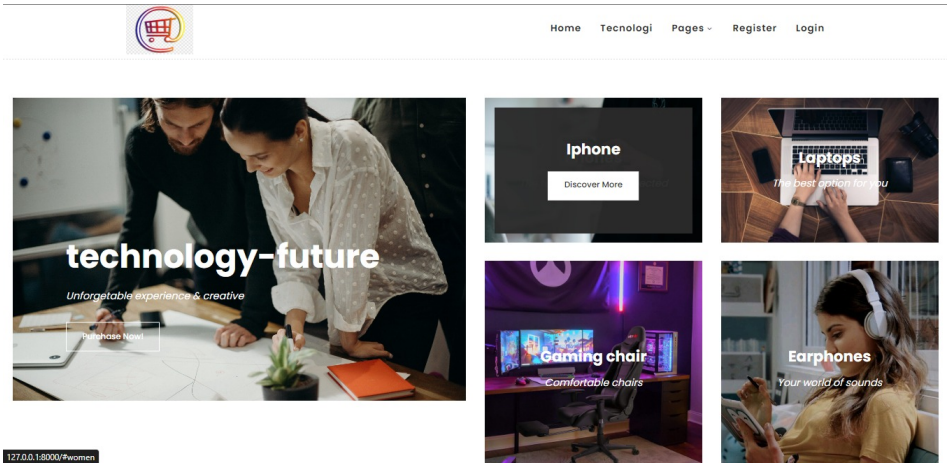
- ¡form action=register"method="POST»: El formulario envía los datos del usuario a la URL - egisterütilizando el método POST. Esto supone que la URL register.[es]tá configurada en las URLs de Django.
- Los campos de entrada (¡input¿) se definen para el primer nombre (first name), el apellido (last

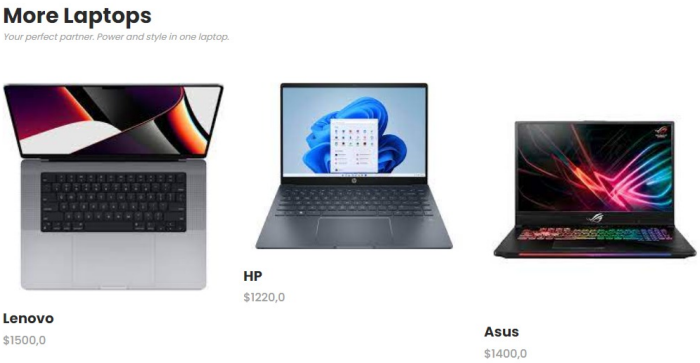
name), el nombre de usuario (username), el correo electrónico (email), la contraseña (password1) y la confirmación de contraseña (password2).

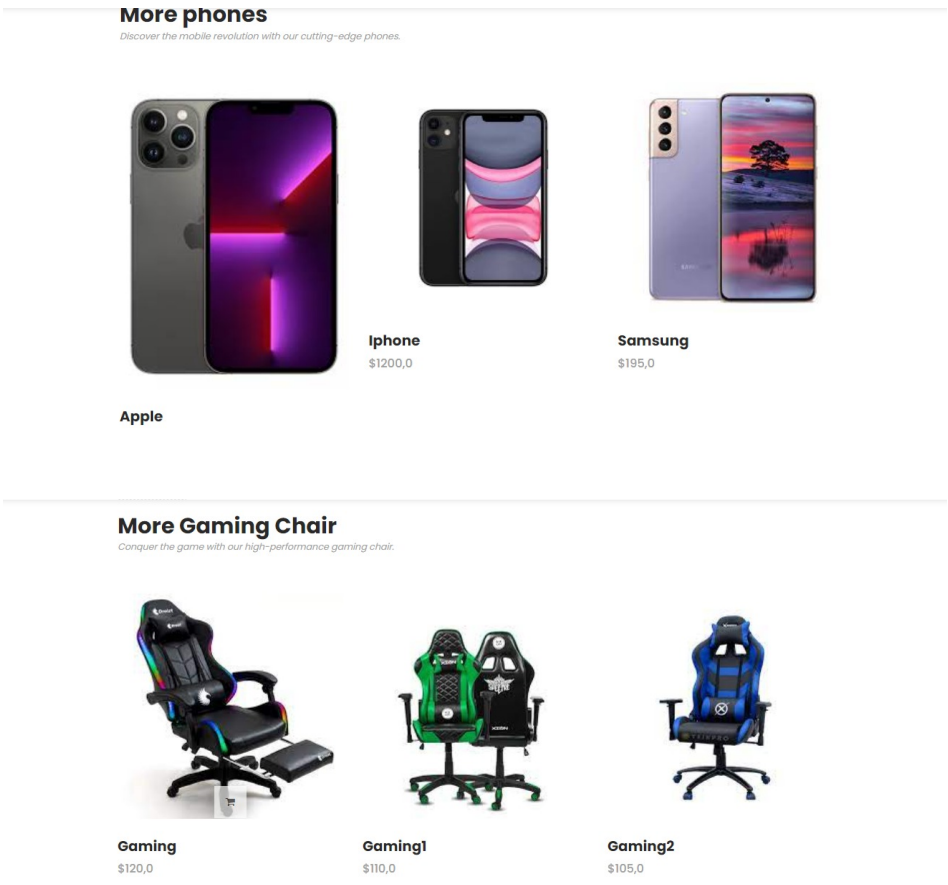
- type="password": El tipo de entrada se establece como "password"para los campos de contraseña, ocultando los caracteres ingresados.
- El ¡div¿se usa para mostrar mensajes que puedan generarse durante el proceso de registro.

3.3. Capturas de la ejecucion de nuestra aplicacion web del proyecto final

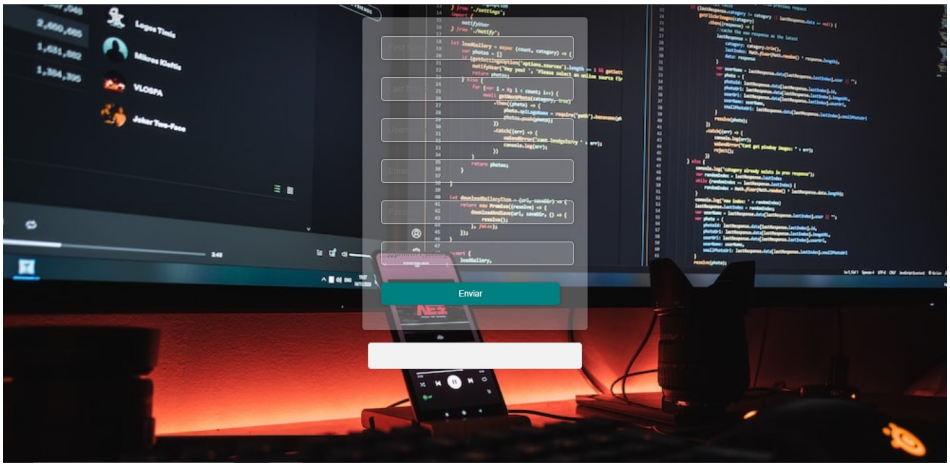
- Secciones de la aplicacion web:



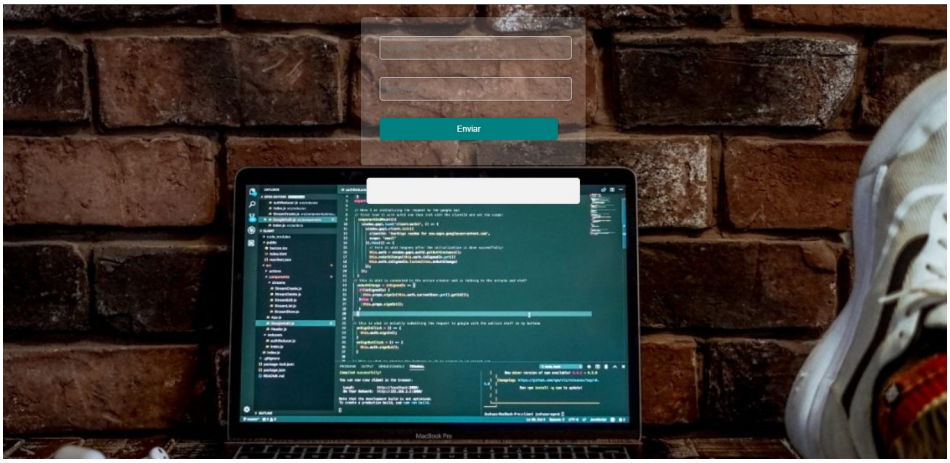




- Register:



- Login:

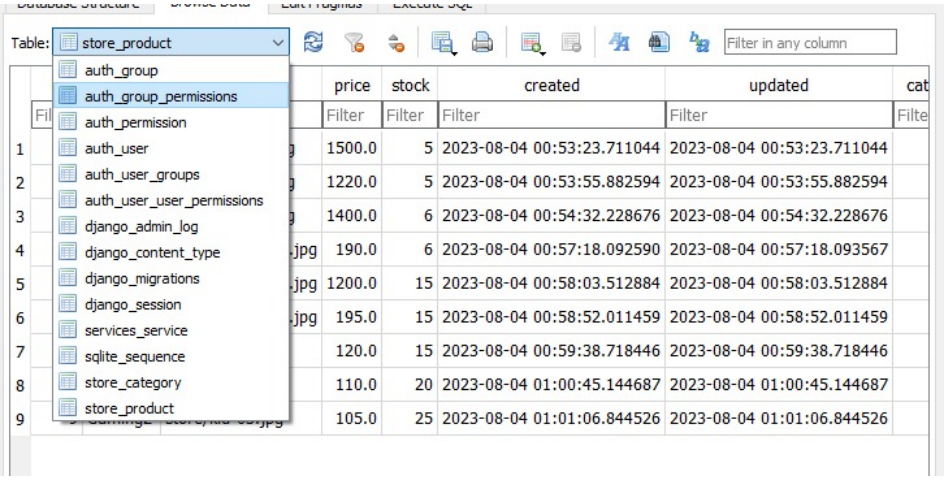


- Base de datos:

	id	name	image	price	stock	created	updated	cat
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filte
1	1	Lenovo	store/men-03.jpg	1500.0	5	2023-08-04 00:53:23.711044	2023-08-04 00:53:23.711044	
2	2	HP	store/men-02.jpg	1220.0	5	2023-08-04 00:53:55.882594	2023-08-04 00:53:55.882594	
3	3	Asus	store/men-01.jpg	1400.0	6	2023-08-04 00:54:32.228676	2023-08-04 00:54:32.228676	
4	4	Apple	store/women-01.jpg	190.0	6	2023-08-04 00:57:18.092590	2023-08-04 00:57:18.093567	
5	5	Iphone	store/women-02.jpg	1200.0	15	2023-08-04 00:58:03.512884	2023-08-04 00:58:03.512884	
6	6	Samsung	store/women-03.jpg	195.0	15	2023-08-04 00:58:52.011459	2023-08-04 00:58:52.011459	
7	7	Gaming	store/kid-01.jpg	120.0	15	2023-08-04 00:59:38.718446	2023-08-04 00:59:38.718446	
8	8	Gaming1	store/kid-02.jpg	110.0	20	2023-08-04 01:00:45.144687	2023-08-04 01:00:45.144687	
9	9	Gaming2	store/kid-03.jpg	105.0	25	2023-08-04 01:01:06.844526	2023-08-04 01:01:06.844526	

	id	name	created	updated
	Filter	Filter	Filter	Filter
1	1	Phones	2023-08-04 00:46:04.932905	2023-08-04 00:46:04.932905
2	2	Laptops	2023-08-04 00:46:14.840152	2023-08-04 00:46:14.840152
3	3	Gaming Chair	2023-08-04 00:46:21.025527	2023-08-04 00:46:21.025527

	id	password	last_login	is_superuser	username	l
	Filter	Filter	Filter	Filter	Filter	F
1	1	pbkdf2_sha256$600000$VQ7vzv9KdlakGao7T0C...	2023-08-04 16:40:51.013166	1	admin	
2	2	pbkdf2_sha256$600000$3MFU732YaBaB1kA1d7T...	2023-08-03 22:32:53.822598	1	admin2	
3	3	pbkdf2_sha256$600000$8YnHIdDEHE1SbvAElcyd...	2023-08-04 03:20:06.800201	0	aaaaaaa	a
4	4	pbkdf2_sha256$600000$EbSnA6lRxUfK0dHS0Szb...	2023-08-04 16:39:54.062439	0	repe	p
5	5	pbkdf2_sha256$600000$8jEZ2WZTsK5SaPhn8a5...	2023-08-04 23:32:43.200375	0	klis	a



3.4. Estructura de laboratorio 09 - Proyecto Final

- El contenido que se entrega del informe del laboratorio 09 - Proyecto Final es el siguiente:

```
Lab09-Informe/
|--- informe-latex
|--- contenido
|   |--- actividades.tex
|   |--- caratula.tex
|   |--- github.tex
|   |--- materiales.tex
|   |--- preguntas.tex
|   |--- referencia.tex
|   |--- rubrica.tex
|   |--- tarea.tex
|--- img
|   |--- logo_abet.png
|   |--- logo_episunsa.png
|   |--- imagen1.jpg
|   |--- imagen2.jpg
|   |--- imagen3.jpg
|   |--- imagen4.jpg
|   |--- imagen5.jpg
|   |--- imagen6.jpg
|   |--- imagen7.jpg
|   |--- imagen8.jpg
|   |--- imagen9.jpg
|   |--- imagen10.jpg
|   |--- imagen11.jpg
|   |--- imagen12.jpg
|--- src
|    |--- admin.py
|    |--- apps.py
|    |--- cart.py
|    |--- context_processor.py
|    |--- forms.py
|    |--- login.txt
|    |--- models.py
```

```
|    |--- register.txt
|    |--- urls.py
|    |--- views.py
|--- pweb2_lab09_proyectofinal.pdf
|--- main.tex
```

4. Pregunta: Como les fue con el trabajo del proyecto final?

- Fue un trabajo interesante para realizarlo con algunas dificultades, pero se pudo terminar. Realizando los objetivos que se pedian en la tarea.

5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	1	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
Total		20		16	

6. Referencias

- https://www.w3schools.com/python/python_reference.asp
- https://docs.djangoproject.com/en/4.2/howto/outputting-pdf/
- https://www.scaler.com/topics/django/relationships-in-django-models/
- https://docs.djangoproject.com/en/4.2/topics/email/