Informe de Laboratorio 09

Tema: Proyecto Final

Nota

Estudiante	Escuela	Asignatura
Sebastian Arley Chirinos Negrón Rodrigo Estefano Viza Cuti Daniel Enrique Marrón Carcausto Diego Renato Llerena Tellez schirinosne@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: I Código: 1702122

Laboratorio	Tema	Duración
09	Proyecto Final	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 31 Julio 2023	Al 07 Agosto 2023

1. Equipos, materiales y temas utilizados

- Programar usando Python.
- Utilización del framework Django para desarrollo web.
- Implementación de plantillas Django para la construcción de interfaces de usuario.
- Uso del Django Rest Framework para crear APIs RESTful.
- Integración de JavaScript para mejorar la interactividad en el frontend.
- Desarrollo de páginas web con HTML y estilización con CSS.
- Empleo de AJAX para realizar peticiones asíncronas en el frontend.
- Consumo de APIs externas, como la API de Google, para la obtención de datos.
- Aplicación de principios de programación orientada a objetos en el diseño del proyecto.
- Separación de intereses en clases: el modelo de datos y su representación visual.
- Utilización de listas y ciclos en el manejo y procesamiento de datos.
- Exploración de conceptos de programación funcional en el desarrollo del proyecto.

2. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/imlosing07/pw2-lab-D-23a.git`

3. Actividades con el repositorio GitHub

3.1. Creacion de de login y register

3.2. forms.py

Este es un script de Python que define dos clases: `SignUpForm` y `LogInForm`. Ambas clases son subclases de formularios de Django, que se utilizan para manejar la entrada del usuario en aplicaciones web. La clase `SignUpForm` es una subclase de `UserCreationForm`, que es un formulario incorporado en Django para crear nuevos usuarios. La clase `LogInForm` es una subclase de `AuthenticationForm`, que es un formulario incorporado en Django para autenticar a usuarios existentes.

La clase `SignUpForm` tiene un método `__init__` que personaliza los campos del formulario para el nombre de usuario, password1 y password2. El método actualiza los atributos del widget para estos campos para agregar clases CSS, establecer el tipo de entrada y agregar texto de marcador de posición. La clase también define dos campos adicionales del formulario: `username` y `email`. La clase `Meta` dentro de la clase `SignUpForm` especifica que el formulario debe usar el modelo incorporado en Django `User` e incluir los campos para el nombre de usuario, password1 y password2.

La clase `LogInForm` también tiene un método `__init__` que personaliza los campos del formulario para el nombre de usuario y la contraseña. Al igual que el `SignUpForm`, este método actualiza los atributos del widget para estos campos. La clase también define un campo adicional del formulario para el nombre de usuario. La clase `Meta` dentro del `LogInForm` especifica que el formulario debe usar el modelo incorporado en Django `User` e incluir los campos para el nombre de usuario y la contraseña.

Listing 1: forms.py

```python
from typing import Any
from django import forms
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth.models import User


class SignUpForm(UserCreationForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['username'].widget.attrs.update({
            'class': 'form-input',
            'required': '',
            'name': 'username',
            'id': 'username',
            'type': 'text',
            'placeholder': 'Nombre de usuario',
            'maxlength': '30',
            'minlength': '6',
        })
        self.fields['password1'].widget.attrs.update({
            'class': 'form-input',
            'required': '',
            'name': 'password1',
            'id': 'password1',
```

```
            'type': 'password',
            'placeholder': 'Contrasea',
            'maxlength': '22',
            'minlength': '8'
        })
        self.fields['password2'].widget.attrs.update({
            'class': 'form-input',
            'required': '',
            'name': 'password2',
            'id': 'password2',
            'type': 'password',
            'placeholder': 'Repita su contrasea',
            'maxlength': '22',
            'minlength': '8'
        })

    username = forms.CharField(max_length=20, label=False)
    email = forms.EmailField(max_length=100)

    class Meta:
        model = User
        fields = ('username', 'password1', 'password2', )


class LogInForm(AuthenticationForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['username'].widget.attrs.update({
            'class': 'form-input',
            'required': '',
            'name': 'username',
            'id': 'username',
            'type': 'text',
            'placeholder': 'Nombre de usuario',
            'maxlength': '30',
            'minlength': '6',
        })
        self.fields['password'].widget.attrs.update({
            'class': 'form-input',
            'required': '',
            'name': 'password1',
            'id': 'password1',
            'type': 'password',
            'placeholder': 'Contrasea',
            'maxlength': '22',
            'minlength': '8'
        })

    username = forms.CharField(max_length=20, label=False)

    class Meta:
        model = User
        fields = ('username', 'password', )
```

3.3. Models.py

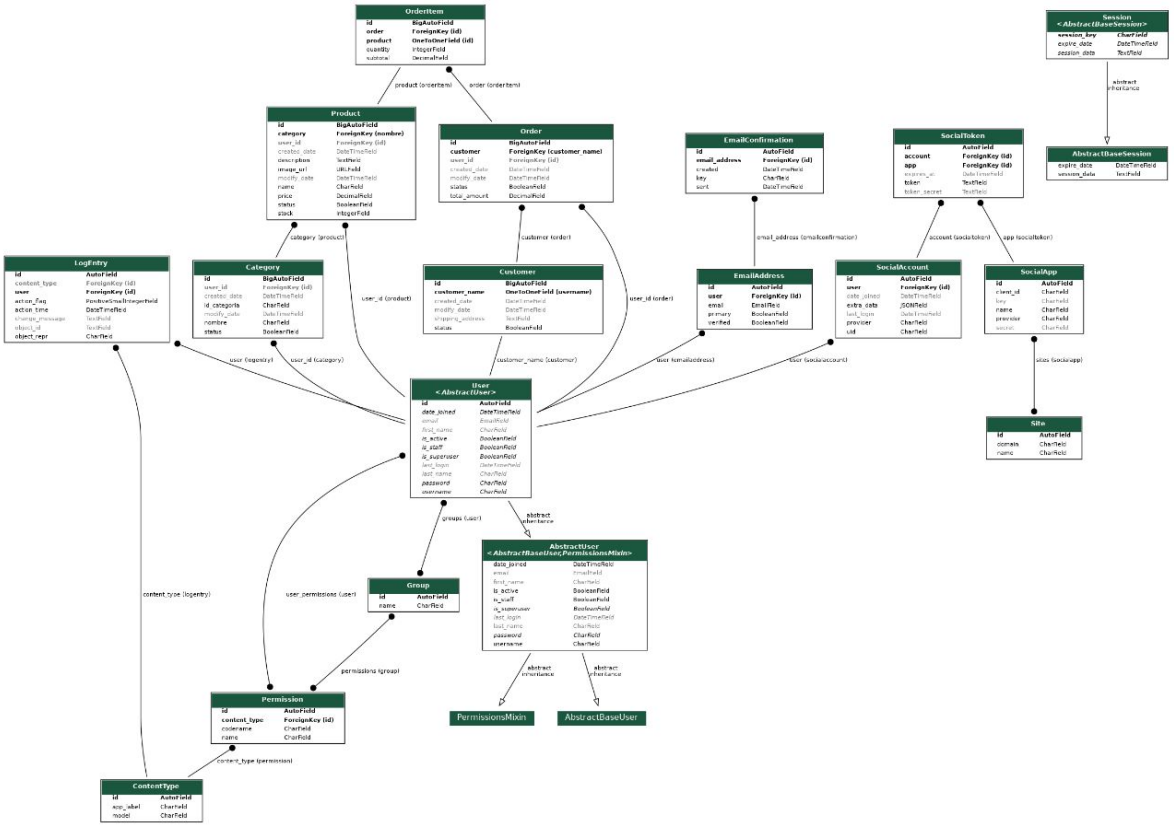
- El modelo `Category` representa una categoría de productos en la tienda. Tiene campos como `nombre`, `id_categoria`, `created_date`, `modify_date`, `status` y `user_id`.
- El modelo `Product` representa un producto en la tienda. Tiene campos como `name`, `description`, `price`, `stock`, `image_url`, `category`, `created_date`, `modify_date`, `status` y `user_id`.
- El modelo `Customer` representa a un cliente en la tienda. Tiene campos como `user`, `shipping_address`, `created_date`, `modify_date`, `status` y `user_id`.
- El modelo `Order` representa a un pedido en la tienda. Tiene campos como `customer`, `items`, `total_amount`, `status`, `created_date`, `modify_date` y `user_id`.

Listing 2: models.py

```python
from django.db import models
from django.contrib.auth.models import User
from tienda.signals import (
    product_updated,
    category_updated,
    order_updated,
    customer_updated,
)
from tienda.middleware import get_current_request

class Category(models.Model):
    nombre = models.CharField(max_length=100,unique=True)
    id_categoria = models.CharField(max_length=100)
    created_date = models.DateTimeField(auto_now_add=True, editable=False)
    modify_date = models.DateTimeField(auto_now=True,editable=False)
    status = models.BooleanField(default=True)
    user_id = models.ForeignKey(User, on_delete=models.CASCADE, to_field='id', null=True,
        blank=True, editable=False)

    def __str__(self):
        return self.nombre

def update_category_user_id(sender, instance, **kwargs):
    request = get_current_request()
    user = request.user if request and request.user.is_authenticated else None
    if user:
        instance.user_id = user.id

category_updated.connect(update_category_user_id, sender=Category)

class Product(models.Model):
    name = models.CharField(max_length=200)
    description = models.TextField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    stock = models.IntegerField()
    image_url = models.URLField()
    category = models.ForeignKey(Category, on_delete=models.CASCADE,to_field='nombre')
    created_date = models.DateTimeField(auto_now_add=True, editable=False)
    modify_date = models.DateTimeField(auto_now=True,editable=False)
    status = models.BooleanField(default=True)
```

```
    user_id = models.ForeignKey(User, on_delete=models.CASCADE, to_field='id', null=True,
        blank=True, editable=False)

    def __str__(self):
        return f"nombre: {self.name} categoria: {self.category}"

def update_product_user_id(sender, instance, **kwargs):
    request = get_current_request()
    user = request.user if request and request.user.is_authenticated else None
    if user:
        instance.user_id = user.id

product_updated.connect(update_product_user_id, sender=Product)

class Customer(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    shipping_address = models.TextField()
    created_date = models.DateTimeField(auto_now_add=True, editable=False)
    modify_date = models.DateTimeField(auto_now=True,editable=False)
    status = models.BooleanField(default=True)
    # Agrega cualquier otro campo adicional que desees para el perfil del cliente
    def __str__(self):
        return self.user.username

def update_customer_user_id(sender, instance, **kwargs):
    request = get_current_request()
    user = request.user if request and request.user.is_authenticated else None
    if user:
        instance.user_id = user.id

customer_updated.connect(update_customer_user_id, sender=Customer)

class Order(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    items = models.ManyToManyField(Product, through='OrderItem')
    total_amount = models.DecimalField(max_digits=10, decimal_places=2)
    status = models.BooleanField(default=True)
    created_date = models.DateTimeField(auto_now_add=True, editable=False)
    modify_date = models.DateTimeField(auto_now_add=True, editable=False)
    user_id = models.ForeignKey(User, on_delete=models.CASCADE, to_field='id', null=True,
        blank=True, editable=False)

def update_order_user_id(sender, instance, **kwargs):
    request = get_current_request()
    user = request.user if request and request.user.is_authenticated else None
    if user:
        instance.user_id = user.id

order_updated.connect(update_order_user_id, sender=Order)

class OrderItem(models.Model):
    product = models.OneToOneField(Product, on_delete=models.CASCADE,to_field='id')
    order = models.ForeignKey(Order, on_delete=models.CASCADE,to_field='id')
    quantity = models.IntegerField()
    subtotal = models.DecimalField(max_digits=10, decimal_places=2)
```

- UML de los modelos:



3.4. views.py

- La vista `home` simplemente devuelve una respuesta que renderiza la plantilla `index.html`.
- La vista `carrito` también devuelve una respuesta que renderiza la plantilla `carrito.html`.
- La vista `signin` maneja el registro de nuevos usuarios en el sitio web. Si la solicitud es un método GET, la vista devuelve una respuesta que renderiza la plantilla `signin.html` con un formulario de registro. Si la solicitud es un método POST, la vista verifica si las contraseñas ingresadas coinciden y, de ser así, intenta crear un nuevo usuario con el nombre de usuario y la contraseña proporcionados. Si el usuario se crea correctamente, se inicia sesión automáticamente y se redirige a la página de inicio. Si ocurre un error al crear el usuario (por ejemplo, si el nombre de usuario ya está en uso), se devuelve una respuesta que renderiza la plantilla `signin.html` con un mensaje de error.
- La vista `user_login` maneja el inicio de sesión de usuarios existentes en el sitio web. Si la solicitud es un método GET, la vista devuelve una respuesta que renderiza la plantilla `login.html` con un formulario de inicio de sesión. Si la solicitud es un método POST, la vista intenta autenticar al usuario con el nombre de usuario y la contraseña proporcionados. Si el usuario se autentica correctamente, se inicia sesión y se redirige a la página de inicio. Si ocurre un error al autenticar al usuario (por ejemplo, si el nombre de usuario o la contraseña son incorrectos), se devuelve una respuesta que renderiza la plantilla `login.html` con un mensaje de error.

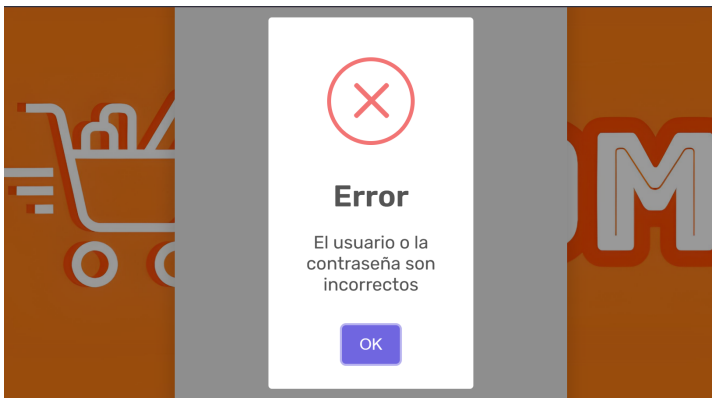
- La vista `user_logout` maneja el cierre de sesión de usuarios en el sitio web. Simplemente cierra la sesión del usuario actual y redirige a la página de inicio.

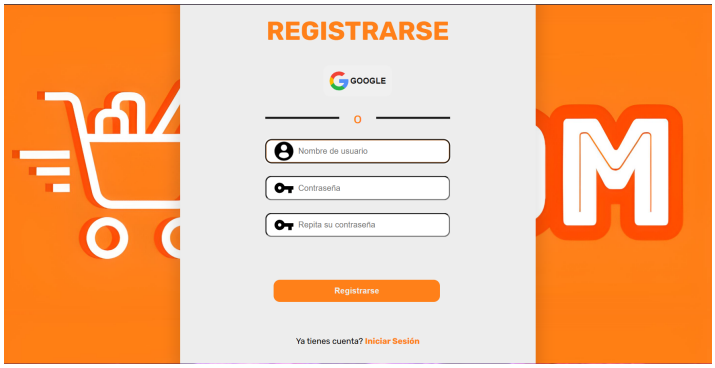
Listing 3: views.py

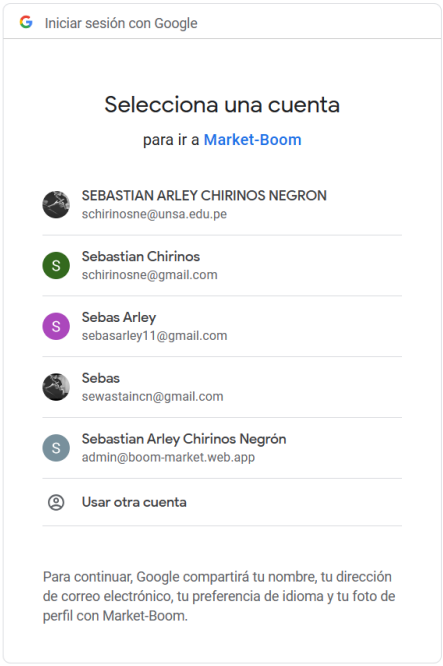
```python
from django.shortcuts import redirect, render
from django.contrib.auth.models import User
from django.contrib.auth import login, logout, authenticate
from django.db import IntegrityError
from .forms import SignUpForm, LogInForm

# Create your views here.


def home(request):
    return render(request, 'index.html')


def carrito(request):
    return render(request, 'carrito.html')


def signin(request):
    if request.method == "GET":
        return render(request, 'signin.html', {
            'form': SignUpForm
        })
    else:
        if request.POST['password1'] == request.POST['password2']:
            try:
                user = User.objects.create_user(
                    username=request.POST['username'],
                    password=request.POST['password1']
                )
                user.save()
                login(request, user,
                      backend='django.contrib.auth.backends.ModelBackend')
                print("se creo la cuenta y se inicio sesion")
                return redirect('home')
            except IntegrityError:
                print("el usuario ya existe")
                return render(request, 'signin.html', {
                    'form': SignUpForm,
                    'error': 'El nombre de usuario no esta disponible',
                }
                )
        print("las contras no coiciden")
        return render(request, 'signin.html', {
            'form': SignUpForm,
            'error': "Las contraseas no coiciden",
        })


def user_login(request):
    if request.method == "GET":
        return render(request, 'login.html', {
```

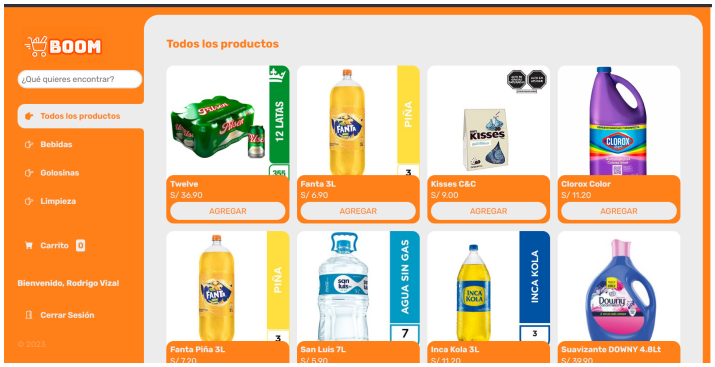
```
            'form': LogInForm
        })
    else:
        user = authenticate(
            request, username=request.POST['username'], password=request.POST['password']
        )
        if user is None:
            print("El usuario o la contrasea son incorrectos")
            return render(request, 'login.html', {
                'form': LogInForm,
                'error': 'El usuario o la contrasea son incorrectos'
            })
        else:
            print('estas siendo redirigido')
            login(request, user)
            return redirect('home')


def user_logout(request):
    logout(request)
    return redirect('home')
```

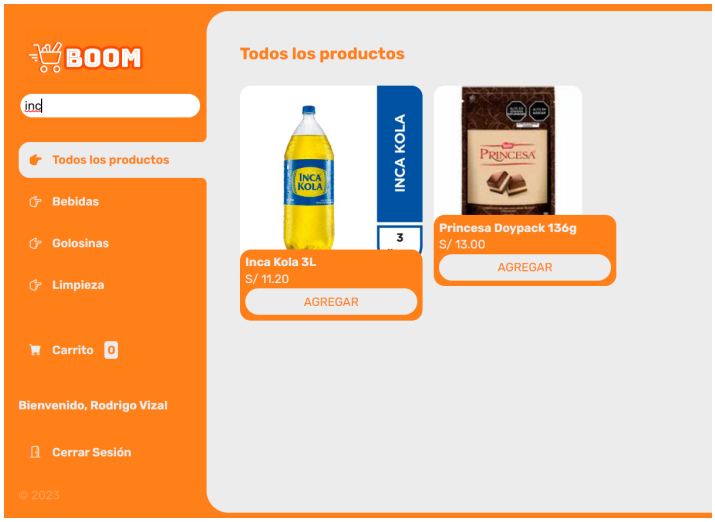
3.5. CAPTURAS DEL Proyecto

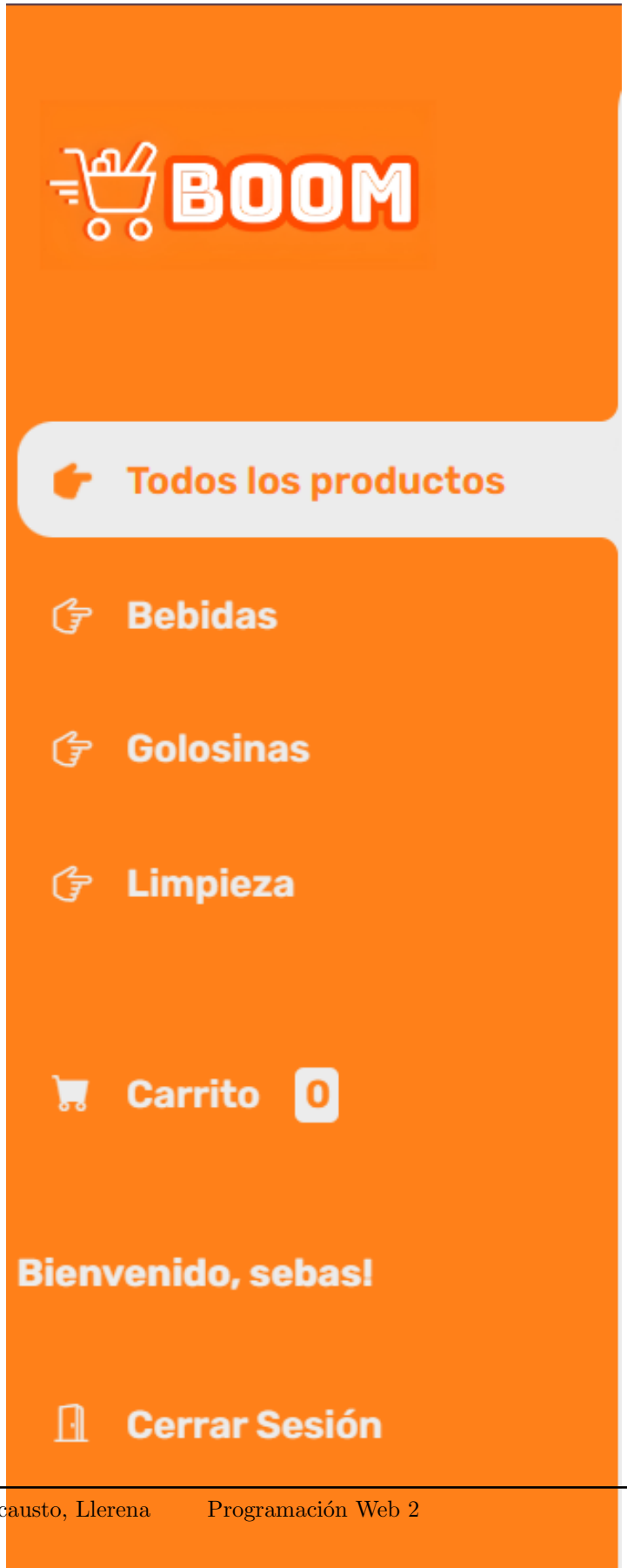


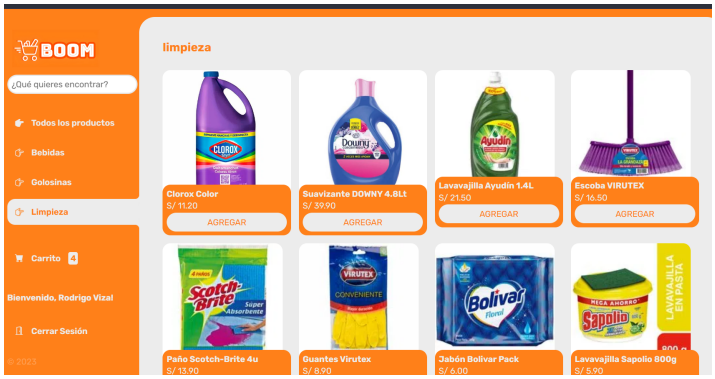

REGISTRARSE
GOOGLE
O
Nombre de usuario
Contraseña
Repita su contraseña
Registrarse
Ya tienes cuenta? Iniciar Sesión


Iniciar sesión con Google
Selecciona una cuenta
para ir a Market-Boom
SEBASTIAN ARLEY CHIRINOS NEGRON
schirinosne@unsa.edu.pe
Sebastian Chirinos
schirinosne@gmail.com
Sebas Arley
sebasarley11@gmail.com
Sebas
sewastaincn@gmail.com
Sebastian Arley Chirinos Negrón
admin@boom-market.web.app
Usar otra cuenta
Para continuar, Google compartirá tu nombre, tu dirección de correo electrónico, tu preferencia de idioma y tu foto de perfil con Market-Boom.


BOOM
¿Qué quieres encontrar?
Todos los productos
Bebidas
Golosinas
Limpieza
Carrito
Bienvenido, Rodrigo Viza!
Cerrar Sesión
Todos los productos
Twelve
S/ 36.90
AGREGAR
Fanta 3L
S/ 6.90
AGREGAR
Kisses C&C
S/ 9.00
AGREGAR
Clorox Color
S/ 11.20
AGREGAR
Fanta Piña 3L
San Luis 7L
Inca Kola 3L
Suavizante DOWNY 4.8Lt


BOOM
inc
Todos los productos
Bebidas
Golosinas
Limpieza
Carrito 0
Bienvenido, Rodrigo Vizal
Cerrar Sesión
© 2023
Todos los productos
INCA KOLA
3
Inca Kola 3L
S/ 11.20
AGREGAR
Princesa Doypack 136g
S/ 13.00
AGREGAR


BOOM
Todos los productos
Bebidas
Golosinas
Limpieza
Carrito
0
Bienvenido, sebas!
Cerrar Sesión


BOOM
¿Qué quieres encontrar?
Todos los productos
Bebidas
Golosinas
Limpieza
Carrito 4
Bienvenido, Rodrigo Vizal
Cerrar Sesión
limpieza
Clorox Color
S/ 11.20
AGREGAR
Suavizante DOWNY 4.8Lt
S/ 39.90
AGREGAR
Lavavajilla Ayudin 1.4L
S/ 21.50
AGREGAR
Escoba VIRUTEX
S/ 16.50
AGREGAR
Paño Scotch-Brite 4u
S/ 13.90
Guantes Virutex
S/ 8.90
Jabón Bolivar Pack
S/ 6.00
Lavavajilla Sapolio 800g
S/ 5.90

3.6. Estructura deL Proyecto final

- El contenido que se entrega en este laboratorio es el siguiente:

```
ProyectoFinal/
|   .gitignore
|   Pweb02_Proyecto-Final.pdf
|   Pweb02_Proyecto-Final.tex
|
+---img
|       1.png
|       2.png
```

```
|       3.png
|       4.png
|       5.png
|       6.png
|       8.png
|       logo_abet.png
|       logo_episunsa.png
|       logo_unsa.jpg
|
\---src
        models.py
        views.py
        forms.py
        serializer.py
```

3.7. 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 1: 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 2: 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	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
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	4	
Total		20		19	

4. Referencias

- `https://www.w3schools.com/python/python_reference.asp`
- `https://docs.python.org/3/tutorial/`