UNIVERSIDAD GERARDO BARRIOS

SAN MIGUEL

Comisión de Acreditación de la Calidad de la Educación Superior
UNIVERSIDAD GERARDO BARRIOS (UGB)
ACREDITADA
2019- 2024

FACULTAD DE CIENCIA Y TECNOLOGIA

Técnico en ingeniería en sistemas y Redes Informáticas

Asignatura: Programación III

Estudiantes: Katerin Michelle Campos Aparicio

Fátima Marisol Batres Santos

Merlín Ibania Diaz Rodríguez

Año: 2024

1. Planteamiento del Problema:
La gestión de una tienda de comida de forma manual presenta varios problemas, como la posibilidad de cometer errores en el registro de inventarios y ventas, dificultad para rastrear productos de alta y baja demanda, y falta de control en la administración de clientes. Estos inconvenientes pueden llevar a una pérdida de ventas y a una gestión ineficiente del negocio.

La solución propuesta consiste en desarrollar un sistema automatizado que permita gestionar inventarios, ventas y la administración de clientes de manera centralizada. Este sistema resolverá los problemas de la gestión manual y permitirá optimizar el control de productos y el seguimiento de las preferencias de los clientes, mejorando la eficiencia operativa de la tienda.

2. Funcionalidad Principal:

Descripción: El proyecto incluye una funcionalidad básica ejecutable para la gestión de productos y ventas, abordando la automatización de procesos en la tienda.

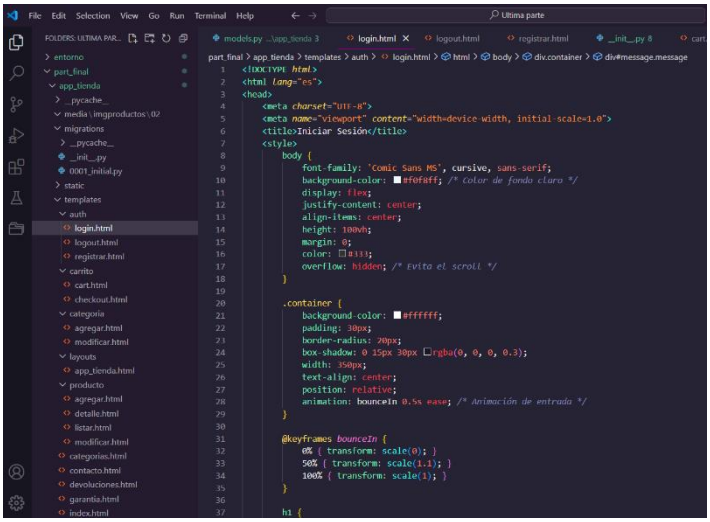
Herramientas Utilizadas:

- **Django** para el backend y la administración general.
- **django-crispy-forms** para mejorar la presentación de formularios.
- **Pillow** para la gestión de imágenes en productos.

Características Principales:

- Registro, edición y eliminación de productos y categorías.
- Administración de usuarios con vistas de login, registro y logout.
- Carrito de compras interactivo y cálculo en tiempo real del total.
- Búsqueda de productos y categorías.

- **Avances del Proyecto: El único avance logrado hasta ahora incluye:**
- Diseño de la página de inicio.
- Implementación de vistas para el inicio de sesión y registro de usuarios. El estado actual del proyecto muestra que se ha comenzado a construir una interfaz amigable para el usuario, pero no se han implementado más funcionalidades.

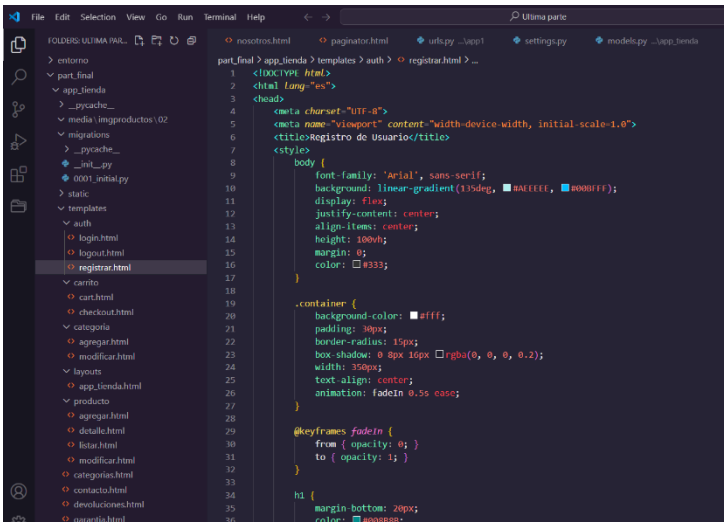


```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Iniciar Sesión</title>
    <style>
        body {
            font-family: 'Comic Sans MS', cursive, sans-serif;
            background-color: #f0f8ff; /* Color de fondo clara */
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            color: #333;
            overflow: hidden; /* Evita el scroll */
        }

        .container {
            background-color: #ffffff;
            padding: 30px;
            border-radius: 20px;
            box-shadow: 0 15px 30px rgba(0, 0, 0, 0.3);
            width: 350px;
            text-align: center;
            position: relative;
            animation: bounceIn 0.5s ease; /* Animación de entrada */
        }

        @keyframes bounceIn {
            0% { transform: scale(0); }
            50% { transform: scale(1.1); }
            100% { transform: scale(1); }
        }

        h1 {
```

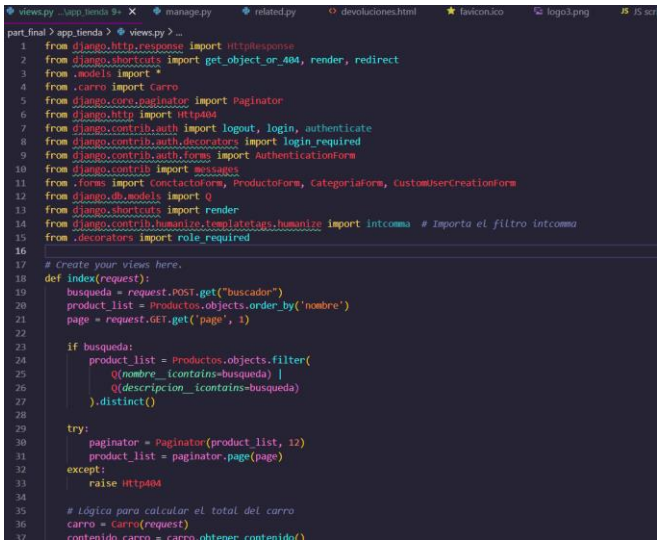


```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registro de Usuario</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background: linear-gradient(135deg, #AEEEEE, #008FFF);
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            color: #333;
        }

        .container {
            background-color: #fff;
            padding: 30px;
            border-radius: 15px;
            box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
            width: 350px;
            text-align: center;
            animation: fadeIn 0.5s ease;
        }

        @keyframes fadeIn {
            from { opacity: 0; }
            to { opacity: 1; }
        }

        h1 {
            margin-bottom: 20px;
```



```
from django.http.response import HttpResponse
from django.shortcuts import get_object_or_404, render, redirect
from .models import *
from .carro import Carro
from django.core.paginator import Paginator
from django.http import Http404
from django.contrib.auth import logout, login, authenticate
from django.contrib.auth.decorators import login_required
from django.contrib.auth.forms import AuthenticationForm
from django.contrib import messages
from .forms import ConctactoForm, ProductoForm, CategoriaForm, CustomUserCreationForm
from django.db.models import Q
from django.shortcuts import render
from django.contrib.humanize.templatetags.humanize import intcomma  # Importa el filtro intcomma
from .decorators import role_required

# Create your views here.
def index(request):
    busqueda = request.POST.get("buscador")
    product_list = Productos.objects.order_by('nombre')
    page = request.GET.get('page', 1)

    if busqueda:
        product_list = Productos.objects.filter(
            Q(nombre__icontains=busqueda) |
            Q(descripcion__icontains=busqueda)
        ).distinct()

    try:
        paginator = Paginator(product_list, 12)
        product_list = paginator.page(page)
    except:
        raise Http404

    # Lógica para calcular el total del carro
    carro = Carro(request)
```

```
    ])
    # Vista de búsqueda
def buscar_producto(request):
    query = request.GET.get('q')  # Obtiene el término de búsqueda desde la URL
    resultados = Productos.objects.filter(
        Q(nombre__icontains=query) |
        Q(descripcion__icontains=query)
    )
    return render(request, 'productos/buscar_resultados.html', {'resultados': resultados, 'query': query})

# Vista de registro de usuario
def registrar(request):
    form = CustomUserCreationForm()
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            messages.success(request, 'Usuario registrado correctamente')
            return redirect('index')
    return render(request, 'auth/registrar.html', {'form': form})

# Vista de inicio de sesión
def login_view(request):
    form = AuthenticationForm()
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)
        if form.is_valid():
            user = form.get_user()
            print(f"Usuario autenticado: {user.username}")
            login(request, user)
            messages.success(request, 'Has iniciado sesión correctamente')
            return redirect('index')
    return render(request, 'auth/login.html', {'form': form})

```

- **Objetivos Faltantes y Plan de Desarrollo:**
- Funcionalidades Pendientes:
- Generación de Reportes (30% completado): Creación de reportes de ventas e inventario con Pandas.
- Notificaciones de Inventario (10% completado): Alertas para inventario bajo y productos próximos a vencer.
- Gestión Completa de Clientes (0% completado): Perfiles detallados de clientes, historial de compras y descuentos personalizados.
- Optimización de la Interfaz: Mejorar la UI en base a la retroalimentación y aplicar el diseño de la plantilla deseada**.**
- **Plan de Desarrollo**:
    - Completar cada componente con tecnologías de apoyo como Django, Pandas, y django-crispy-forms.
    - Implementar un sistema de notificaciones y reportes antes de la integración final de gestión de clientes.