Sistema de Seguridad Portuaria - Puerto Seguro

Documentación Técnica Completa

Versión: 1.0
Fecha: 30 de Diciembre, 2025
Tecnologías: Python 3.13, Django 6.0, uv, Bootstrap 5
Autor: Proyecto de Vinculación ULEAM

Tabla de Contenidos

---

1. Introducción

Este proyecto implementa una **Mini Demo de Seguridad Portuaria** que demuestra los conceptos fundamentales de:

- **Autenticación (Authentication):** Verificar la identidad del usuario (Login/Logout)
- **Autorización (Authorization):** Control de acceso basado en roles
- **Auditoría:** Registro de quién realizó cada acción
- **Protección CSRF:** Defensa contra ataques Cross-Site Request Forgery

Roles del Sistema

Rol	Descripción	Permisos
Admin	Jefe de Puerto	Acceso total (CRUD completo)

Rol	Descripción	Permisos
Operador	Registra Barcos	Ver, Crear, Editar
Guardia	Solo Lectura	Solo Ver

---

2. Requisitos Previos

Software Necesario

- **Python 3.11+** (recomendado 3.13)
- **uv** - Gestor de paquetes moderno para Python
- **Git** (opcional, para control de versiones)

Instalación de uv

```
# En Windows (PowerShell como Administrador)
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"

# En Linux/macOS
curl -LsSf https://astral.sh/uv/install.sh | sh
```

Verificar instalación:

```
uv --version
```

---

3. Fase 1: Configuración del Entorno con uv

3.1 Crear el Directorio del Proyecto

```
# Navegar a tu directorio de trabajo
cd C:\Users\TuUsuario\Documents\proyectos

# Crear directorio del proyecto
mkdir puerto_seguro
cd puerto_seguro
```

3.2 Inicializar Proyecto con uv

```
# Inicializar proyecto (crea pyproject.toml, README.md, etc.)
uv init
```

Salida esperada:

```
Initialized project `puerto-seguro`
```

3.3 Instalar Django

```
# uv instala Django y crea el entorno virtual automáticamente
uv add django
```

Salida esperada:

```
Using CPython 3.13.x
Creating virtual environment at: .venv
Resolved 5 packages in XXXms
Installed 4 packages in XXXs
 + asgiref==3.11.0
 + django==6.0
 + sqlparse==0.5.5
 + tzdata==2025.3
```

3.4 Crear el Proyecto Django

```
# Crear proyecto Django llamado 'config' en el directorio actual
uv run django-admin startproject config .
```

3.5 Crear la Aplicación Principal

```
# Crear la app 'core' que contendrá toda la lógica
uv run python manage.py startapp core
```

Estructura resultante:

```
puerto_seguro/
   .venv/              # Entorno virtual (creado por uv)
   config/             # Configuración del proyecto Django
       __init__.py
       asgi.py
       settings.py
       urls.py
       wsgi.py
   core/               # Aplicación principal
       __init__.py
       admin.py
       apps.py
       models.py
       tests.py
       views.py
   manage.py
   pyproject.toml
   README.md
```

---

4. Fase 2: Modelado de Datos

4.1 Modelo de Usuario Personalizado

> **IMPORTANTE:** El modelo de usuario personalizado DEBE definirse ANTES de ejecutar la primera migración. Esta es la "Regla de Oro de Django".

Archivo: `core/models.py`

```
"""
Modelos del Sistema de Seguridad Portuaria
==========================================
Implementa el modelo de Usuario personalizado con roles y el modelo Barco.
"""

from django.db import models
from django.contrib.auth.models import AbstractUser


class Usuario(AbstractUser):
    """
    Usuario personalizado con sistema de roles.

    Extiende AbstractUser para añadir el campo 'rol' que determina
    los permisos del usuario en el sistema.

    Roles disponibles:
    - admin: Administrador (Jefe de Puerto) - Acceso total
    - operador: Operador - Puede registrar y gestionar barcos
    - guardia: Guardia - Solo puede ver información (lectura)
    """
    ROLES = (
        ('admin', 'Administrador (Jefe de Puerto)'),
        ('operador', 'Operador (Registra Barcos)'),
        ('guardia', 'Guardia (Solo ve accesos)'),
    )

    rol = models.CharField(
        max_length=20,
        choices=ROLES,
        default='guardia',
        verbose_name='Rol del Usuario',
        help_text='Define los permisos del usuario en el sistema'
    )

    class Meta:
```

```
        verbose_name = 'Usuario'
        verbose_name_plural = 'Usuarios'

    def __str__(self):
        return f"{self.username} ({self.get_rol_display()})"


class Barco(models.Model):
    """
    Modelo que representa un barco registrado en el puerto.

    Incluye información básica del barco y datos de auditoría
    para saber quién y cuándo registró cada embarcación.
    """
    nombre = models.CharField(
        max_length=100,
        verbose_name='Nombre del Barco'
    )
    imo = models.CharField(
        max_length=20,
        unique=True,
        verbose_name='Número IMO',
        help_text='Número de identificación marítima internacional (único)'
    )
    bandera = models.CharField(
        max_length=50,
        verbose_name='País de Bandera',
        default='Ecuador'
    )
    tipo = models.CharField(
        max_length=50,
        choices=[
            ('carga', 'Buque de Carga'),
            ('pasajeros', 'Buque de Pasajeros'),
            ('petrolero', 'Petrolero'),
            ('pesquero', 'Pesquero'),
            ('otro', 'Otro'),
        ],
        default='carga',
        verbose_name='Tipo de Embarcación'
    )
    fecha_llegada = models.DateTimeField(
        auto_now_add=True,
        verbose_name='Fecha de Registro'
    )
```

```python
    # Auditoría: Quién registró el barco (Best Practice)
    registrado_por = models.ForeignKey(
        Usuario,
        on_delete=models.SET_NULL,
        null=True,
        verbose_name='Registrado por',
        related_name='barcos_registrados'
    )

    class Meta:
        verbose_name = 'Barco'
        verbose_name_plural = 'Barcos'
        ordering = ['-fecha_llegada']

    def __str__(self):
        return f"{self.nombre} (IMO: {self.imo})"
```

4.2 Explicación de los Campos

Modelo Usuario

Campo	Tipo	Descripción
`rol`	CharField	Define el nivel de acceso del usuario
(heredados)	-	username, email, password, first_name, last_name, etc.

Modelo Barco

Campo	Tipo	Descripción
`nombre`	CharField	Nombre de la embarcación
`imo`	CharField	Número IMO único (identificador internacional)
`bandera`	CharField	País de registro del barco
`tipo`	CharField	Tipo de embarcación (carga, pasajeros, etc.)
`fecha_llegada`	DateTimeField	Fecha/hora de registro (automática)
`registrado_por`	ForeignKey	Usuario que registró el barco (auditoría)

---

5. Fase 3: Configuración del Proyecto

5.1 Modificar settings.py

Archivo: `config/settings.py`

Registrar la aplicación core Buscar `INSTALLED_APPS` y agregar `'core'`:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Apps del proyecto
    'core',
]
```

Configurar directorio de templates Buscar `TEMPLATES` y modificar `'DIRS'`:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],  # Directorio global de templates
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Configurar idioma y zona horaria

```
LANGUAGE_CODE = 'es-ec'

TIME_ZONE = 'America/Guayaquil'
```

Configurar modelo de usuario y autenticación Agregar al final del archivo:

```
# ================================================================================
# CONFIGURACIÓN DEL SISTEMA DE SEGURIDAD PORTUARIA
# ================================================================================

# Modelo de Usuario Personalizado (CRÍTICO: debe configurarse antes de migraciones)
AUTH_USER_MODEL = 'core.Usuario'
```

```python
# Configuración de Autenticación
LOGIN_REDIRECT_URL = 'dashboard'  # Después de login exitoso
LOGOUT_REDIRECT_URL = 'home'      # Después de logout
LOGIN_URL = 'login'               # Redirige aquí si no está autenticado
```

---

6. Fase 4: Panel de Administración

6.1 Configurar admin.py

Archivo: `core/admin.py`

```python
"""
Configuración del Panel de Administración
==========================================
Personaliza la interfaz de admin para gestionar usuarios y barcos.
"""

from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from .models import Usuario, Barco


@admin.register(Usuario)
class UsuarioAdmin(UserAdmin):
    """
    Admin personalizado para el modelo Usuario.
    Extiende UserAdmin para mantener la funcionalidad de gestión de contraseñas.
    """
    list_display = ('username', 'email', 'rol', 'is_active', 'is_staff')
    list_filter = ('rol', 'is_active', 'is_staff')
    search_fields = ('username', 'email', 'first_name', 'last_name')
    ordering = ('username',)

    # Añadir el campo 'rol' al formulario de edición
    fieldsets = UserAdmin.fieldsets + (
        ('Rol en el Sistema', {'fields': ('rol',)}),
    )

    # Añadir el campo 'rol' al formulario de creación
    add_fieldsets = UserAdmin.add_fieldsets + (
        ('Rol en el Sistema', {'fields': ('rol',)}),
    )


@admin.register(Barco)
```

```python
class BarcoAdmin(admin.ModelAdmin):
    """
    Admin para gestionar los registros de barcos.
    """
    list_display = ('nombre', 'imo', 'bandera', 'tipo', 'fecha_llegada', 'registrado_por')
    list_filter = ('tipo', 'bandera', 'fecha_llegada')
    search_fields = ('nombre', 'imo')
    readonly_fields = ('fecha_llegada', 'registrado_por')
    ordering = ('-fecha_llegada',)

    def save_model(self, request, obj, form, change):
        """Guarda automáticamente quién registró el barco desde el admin."""
        if not change:  # Solo en creación
            obj.registrado_por = request.user
        super().save_model(request, obj, form, change)
```

---

7. Fase 5: Formularios

7.1 Crear forms.py

Archivo: `core/forms.py` (crear nuevo archivo)

```python
"""
Formularios del Sistema de Seguridad Portuaria
==============================================
Define los formularios para la captura de datos.
"""

from django import forms
from .models import Barco


class BarcoForm(forms.ModelForm):
    """
    Formulario para crear y editar barcos.

    Usa ModelForm para generar automáticamente los campos
    basándose en el modelo Barco.
    """

    class Meta:
        model = Barco
        fields = ['nombre', 'imo', 'bandera', 'tipo']
        widgets = {
            'nombre': forms.TextInput(attrs={
```

```
            'class': 'form-control',
            'placeholder': 'Ej: MSC Esperanza'
        }),
        'imo': forms.TextInput(attrs={
            'class': 'form-control',
            'placeholder': 'Ej: 9484525'
        }),
        'bandera': forms.TextInput(attrs={
            'class': 'form-control',
            'placeholder': 'Ej: Panamá'
        }),
        'tipo': forms.Select(attrs={
            'class': 'form-select'
        }),
    }
    help_texts = {
        'imo': 'Número único de 7 dígitos asignado por la OMI',
    }
```

---

8. Fase 6: Vistas y Lógica de Seguridad

8.1 Implementar views.py

Archivo: `core/views.py`

```
"""
Vistas del Sistema de Seguridad Portuaria
==========================================
Implementa autenticación y autorización basada en roles.
"""

from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required, user_passes_test
from django.contrib import messages
from .models import Barco
from .forms import BarcoForm


# ============================================================================
# FUNCIONES DE VERIFICACIÓN DE ROLES (Autorización)
# ============================================================================

def es_admin(user):
    """Verifica si el usuario es administrador."""
    return user.rol == 'admin' or user.is_superuser
```

```python
def es_operador(user):
    """Verifica si el usuario es operador o tiene permisos superiores."""
    return user.rol in ('operador', 'admin') or user.is_superuser


def es_guardia_o_superior(user):
    """Verifica si el usuario tiene al menos rol de guardia (cualquier rol)."""
    return user.rol in ('guardia', 'operador', 'admin') or user.is_superuser


# ============================================================================
# VISTAS PÚBLICAS
# ============================================================================

def home(request):
    """
    Página principal del sistema.
    Muestra información diferente según el estado de autenticación.
    """
    context = {
        'total_barcos': Barco.objects.count(),
    }
    return render(request, 'home.html', context)


# ============================================================================
# VISTAS PROTEGIDAS (Requieren Login)
# ============================================================================

@login_required
def lista_barcos(request):
    """
    Lista todos los barcos registrados.
    Requiere: Usuario autenticado (cualquier rol).
    """
    barcos = Barco.objects.all().select_related('registrado_por')
    return render(request, 'lista_barcos.html', {'barcos': barcos})


@login_required
def detalle_barco(request, pk):
    """
    Muestra el detalle de un barco específico.
    Requiere: Usuario autenticado (cualquier rol).
```

```
    """
    from django.shortcuts import get_object_or_404
    barco = get_object_or_404(Barco, pk=pk)
    return render(request, 'detalle_barco.html', {'barco': barco})


# ============================================================================
# VISTAS RESTRINGIDAS (Requieren Login + Rol Específico)
# ============================================================================

@login_required
@user_passes_test(es_operador, login_url='home')
def crear_barco(request):
    """
    Formulario para registrar un nuevo barco.
    Requiere: Usuario autenticado con rol 'operador' o 'admin'.

    El decorador @user_passes_test verifica el rol antes de permitir acceso.
    Si el usuario no tiene permisos, es redirigido a 'home'.
    """
    if request.method == 'POST':
        form = BarcoForm(request.POST)
        if form.is_valid():
            barco = form.save(commit=False)
            barco.registrado_por = request.user  # Auditoría: guardamos quién creó
            barco.save()
            messages.success(request, f' Barco "{barco.nombre}" registrado exitosamente.')
            return redirect('lista_barcos')
    else:
        form = BarcoForm()

    return render(request, 'crear_barco.html', {'form': form})


@login_required
@user_passes_test(es_operador, login_url='home')
def editar_barco(request, pk):
    """
    Formulario para editar un barco existente.
    Requiere: Usuario autenticado con rol 'operador' o 'admin'.
    """
    from django.shortcuts import get_object_or_404
    barco = get_object_or_404(Barco, pk=pk)

    if request.method == 'POST':
        form = BarcoForm(request.POST, instance=barco)
```

```
            if form.is_valid():
                form.save()
                messages.success(request, f' Barco "{barco.nombre}" actualizado.')
                return redirect('lista_barcos')
        else:
            form = BarcoForm(instance=barco)

        return render(request, 'editar_barco.html', {'form': form, 'barco': barco})


    @login_required
    @user_passes_test(es_admin, login_url='home')
    def eliminar_barco(request, pk):
        """
        Elimina un barco del sistema.
        Requiere: Usuario autenticado con rol 'admin' únicamente.
        """
        from django.shortcuts import get_object_or_404
        barco = get_object_or_404(Barco, pk=pk)

        if request.method == 'POST':
            nombre = barco.nombre
            barco.delete()
            messages.warning(request, f' Barco "{nombre}" eliminado del sistema.')
            return redirect('lista_barcos')

        return render(request, 'eliminar_barco.html', {'barco': barco})


    # ============================================================================
    # VISTA DEL PANEL DE CONTROL (Dashboard)
    # ============================================================================

    @login_required
    def dashboard(request):
        """
        Panel de control con estadísticas del sistema.
        Muestra información según el rol del usuario.
        """
        from django.db.models import Count

        context = {
            'total_barcos': Barco.objects.count(),
            'barcos_por_tipo': Barco.objects.values('tipo').annotate(total=Count('tipo')),
        }
```

```
    # Los operadores/admins ven sus propios registros
    if es_operador(request.user):
        context['mis_registros'] = Barco.objects.filter(
            registrado_por=request.user
        ).count()

    return render(request, 'dashboard.html', context)
```

8.2 Explicación de los Decoradores de Seguridad

Decorador	Función	Ejemplo
`@login_required`	Requiere que el usuario esté autenticado	Protege vistas básicas
`@user_passes_test(func)`	Ejecuta una función que retorna True/False	Control de roles

Flujo de autorización:

```
Usuario solicita /barcos/nuevo/



 @login_required     ¿Está logueado?

         No → Redirige a /login/
         Sí

 @user_passes_test(es_operador)  ¿Es operador o admin?

         No → Redirige a home
         Sí

   Vista ejecutada
```

---

9. Fase 7: Configuración de URLs

9.1 Modificar urls.py

Archivo: `config/urls.py`

```
"""
URL configuration for config project.
"""
```

```python
from django.contrib import admin
from django.urls import path, include
from core import views

urlpatterns = [
    # Panel de Administración de Django
    path('admin/', admin.site.urls),

    # Sistema de Autenticación de Django (login, logout, password reset, etc.)
    path('accounts/', include('django.contrib.auth.urls')),

    # Páginas Públicas
    path('', views.home, name='home'),

    # Panel de Control
    path('dashboard/', views.dashboard, name='dashboard'),

    # Gestión de Barcos (CRUD)
    path('barcos/', views.lista_barcos, name='lista_barcos'),
    path('barcos/nuevo/', views.crear_barco, name='crear_barco'),
    path('barcos/<int:pk>/', views.detalle_barco, name='detalle_barco'),
    path('barcos/<int:pk>/editar/', views.editar_barco, name='editar_barco'),
    path('barcos/<int:pk>/eliminar/', views.eliminar_barco, name='eliminar_barco'),
]
```

9.2 URLs Disponibles

URL	Vista	Acceso
/	home	Público
/accounts/login/	Login de Django	Público
/accounts/logout/	Logout	Autenticados
/dashboard/	dashboard	Autenticados
/barcos/	lista_barcos	Autenticados
/barcos/nuevo/	crear_barco	Operador, Admin
/barcos/<pk>/	detalle_barco	Autenticados
/barcos/<pk>/editar/	editar_barco	Operador, Admin
/barcos/<pk>/eliminar/	eliminar_barco	Solo Admin
/admin/	Panel Admin	Staff

10. Fase 8: Templates (Interfaz de Usuario)

10.1 Crear estructura de carpetas

```
mkdir templates
mkdir templates\registration
```

Estructura de templates:

```
templates/
   base.html               # Plantilla base con navbar
   home.html               # Página de inicio
   dashboard.html          # Panel de control
   lista_barcos.html       # Lista de barcos
   crear_barco.html        # Formulario de creación
   detalle_barco.html      # Detalle de un barco
   editar_barco.html       # Formulario de edición
   eliminar_barco.html     # Confirmación de eliminación
   registration/
       login.html          # Página de login
```

10.2 Template Base (base.html)

Archivo: `templates/base.html`

```
{% load static %}
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>{% block title %}Puerto Seguro{% endblock %}</title>

    <!-- Bootstrap 5 CSS -->
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
      rel="stylesheet"
    />
    <!-- Bootstrap Icons -->
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css"
      rel="stylesheet"
    />

    <style>
      :root {
        --primary-color: #0d6efd;
        --secondary-color: #6c757d;
```

```
  --success-color: #198754;
  --warning-color: #ffc107;
  --danger-color: #dc3545;
}

body {
  min-height: 100vh;
  display: flex;
  flex-direction: column;
}

.navbar-brand {
  font-weight: bold;
}

.navbar-brand i {
  color: #ffc107;
}

main {
  flex: 1;
}

.hero-section {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  padding: 4rem 0;
}

.card-role {
  border-left: 4px solid var(--primary-color);
}

.card-role.admin {
  border-left-color: var(--danger-color);
}
.card-role.operador {
  border-left-color: var(--success-color);
}
.card-role.guardia {
  border-left-color: var(--warning-color);
}

.badge-admin {
  background-color: var(--danger-color);
}
```

```
    .badge-operador {
      background-color: var(--success-color);
    }
    .badge-guardia {
      background-color: var(--warning-color);
      color: #000;
    }

    footer {
      background-color: #212529;
      color: #6c757d;
      padding: 1rem 0;
    }

    .table-hover tbody tr:hover {
      background-color: rgba(13, 110, 253, 0.1);
    }
  </style>

  {% block extra_css %}{% endblock %}
</head>
<body>
  <!-- Barra de Navegación -->
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
      <a class="navbar-brand" href="{% url 'home' %}">
        <i class="bi bi-ship"></i> Puerto Seguro
      </a>

      <button
        class="navbar-toggler"
        type="button"
        data-bs-toggle="collapse"
        data-bs-target="#navbarNav"
      >
        <span class="navbar-toggler-icon"></span>
      </button>

      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav me-auto">
          <li class="nav-item">
            <a class="nav-link" href="{% url 'home' %}">
              <i class="bi bi-house"></i> Inicio
            </a>
          </li>
```

```
    {% if user.is_authenticated %}
    <li class="nav-item">
      <a class="nav-link" href="{% url 'dashboard' %}">
        <i class="bi bi-speedometer2"></i> Dashboard
      </a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{% url 'lista_barcos' %}">
        <i class="bi bi-list-ul"></i> Ver Barcos
      </a>
    </li>

    {# Solo operadores y admins ven el botón de registrar #} {% if
    user.rol == 'operador' or user.rol == 'admin' or user.is_superuser
    %}
    <li class="nav-item">
      <a class="nav-link text-success" href="{% url 'crear_barco' %}">
        <i class="bi bi-plus-circle"></i> Registrar Barco
      </a>
    </li>
    {% endif %} {% endif %}
  </ul>

  <ul class="navbar-nav">
    {% if user.is_authenticated %}
    <li class="nav-item dropdown">
      <a
        class="nav-link dropdown-toggle"
        href="#"
        id="userDropdown"
        role="button"
        data-bs-toggle="dropdown"
        aria-expanded="false"
      >
        <i class="bi bi-person-circle"></i>
        {{ user.username }}
        <span class="badge badge-{{ user.rol }}">
          {{ user.get_rol_display }}
        </span>
      </a>
      <ul
        class="dropdown-menu dropdown-menu-end"
        aria-labelledby="userDropdown"
      >
        <li>
          <span class="dropdown-item-text">
```

```
                <small class="text-muted">Conectado como:</small><br />
                <strong
                  >{{ user.get_full_name|default:user.username }}</strong
                >
              </span>
            </li>
            <li><hr class="dropdown-divider" /></li>
            {% if user.is_staff %}
            <li>
              <a class="dropdown-item" href="{% url 'admin:index' %}">
                <i class="bi bi-gear"></i> Panel Admin
              </a>
            </li>
            {% endif %}
            <li>
              <form
                action="{% url 'logout' %}"
                method="post"
                class="d-inline"
              >
                {% csrf_token %}
                <button type="submit" class="dropdown-item text-danger">
                  <i class="bi bi-box-arrow-right"></i> Cerrar Sesión
                </button>
              </form>
            </li>
          </ul>
        </li>
        {% else %}
        <li class="nav-item">
          <a class="nav-link" href="{% url 'login' %}">
            <i class="bi bi-box-arrow-in-right"></i> Iniciar Sesión
          </a>
        </li>
        {% endif %}
      </ul>
    </div>
  </div>
</nav>

<!-- Mensajes Flash -->
{% if messages %}
<div class="container mt-3">
  {% for message in messages %}
  <div
    class="alert alert-{{ message.tags }} alert-dismissible fade show"
```

```
        role="alert"
      >
        {{ message }}
        <button
          type="button"
          class="btn-close"
          data-bs-dismiss="alert"
          aria-label="Close"
        ></button>
      </div>
      {% endfor %}
    </div>
    {% endif %}

    <!-- Contenido Principal -->
    <main>{% block content %}{% endblock %}</main>

    <!-- Footer -->
    <footer class="mt-auto">
      <div class="container text-center">
        <small>
          <i class="bi bi-shield-lock"></i>
          Sistema de Seguridad Portuaria | ULEAM - Vinculación 2025
        </small>
      </div>
    </footer>

    <!-- Bootstrap 5 JS Bundle -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.j

    {% block extra_js %}{% endblock %}
  </body>
</html>
```

10.3 Template de Login (registration/login.html)

Archivo: `templates/registration/login.html`

```
{% extends 'base.html' %} {% block title %}Iniciar Sesión - Puerto Seguro{%
endblock %} {% block content %}
<div class="container py-5">
  <div class="row justify-content-center">
    <div class="col-md-5">
      <div class="card shadow">
        <div class="card-header bg-primary text-white text-center">
          <h4 class="mb-0"><i class="bi bi-shield-lock"></i> Identificación</h4>
```

```
</div>
<div class="card-body p-4">
  <p class="text-muted text-center mb-4">
    Ingrese sus credenciales para acceder al Sistema de Control
    Portuario
  </p>

  {% if form.errors %}
  <div class="alert alert-danger">
    <i class="bi bi-exclamation-triangle"></i>
    Usuario o contraseña incorrectos. Por favor, intente nuevamente.
  </div>
  {% endif %}

  <form method="post">
    {% csrf_token %}

    <div class="mb-3">
      <label for="id_username" class="form-label">
        <i class="bi bi-person"></i> Usuario
      </label>
      <input
        type="text"
        name="username"
        id="id_username"
        class="form-control form-control-lg"
        placeholder="Ingrese su usuario"
        autofocus
        required
      />
    </div>

    <div class="mb-4">
      <label for="id_password" class="form-label">
        <i class="bi bi-key"></i> Contraseña
      </label>
      <input
        type="password"
        name="password"
        id="id_password"
        class="form-control form-control-lg"
        placeholder="Ingrese su contraseña"
        required
      />
    </div>
```

```
      <div class="d-grid">
        <button type="submit" class="btn btn-primary btn-lg">
          <i class="bi bi-box-arrow-in-right"></i> Ingresar al Sistema
        </button>
      </div>
    </form>
  </div>
  <div class="card-footer text-center text-muted">
    <small>
      <i class="bi bi-info-circle"></i>
      Si olvidó su contraseña, contacte al administrador
    </small>
  </div>
</div>

<!-- Información de Demo -->
<div class="card mt-4 border-info">
  <div class="card-header bg-info text-white">
    <i class="bi bi-lightbulb"></i> Usuarios de Prueba
  </div>
  <div class="card-body">
    <table class="table table-sm table-borderless mb-0">
      <thead>
        <tr>
          <th>Usuario</th>
          <th>Rol</th>
          <th>Permisos</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td><code>admin</code></td>
          <td><span class="badge bg-danger">Admin</span></td>
          <td>Acceso total</td>
        </tr>
        <tr>
          <td><code>ana_operadora</code></td>
          <td><span class="badge bg-success">Operador</span></td>
          <td>Ver + Registrar</td>
        </tr>
        <tr>
          <td><code>pepe_guardia</code></td>
          <td><span class="badge bg-warning text-dark">Guardia</span></td>
          <td>Solo ver</td>
        </tr>
      </tbody>
```

```
          </table>
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

10.4 Otros Templates

Los siguientes templates están disponibles en el proyecto:

- **home.html**: Página de inicio con información del sistema y roles
- **dashboard.html**: Panel de control con estadísticas
- **lista_barcos.html**: Tabla con todos los barcos registrados
- **crear_barco.html**: Formulario para registrar nuevos barcos
- **detalle_barco.html**: Vista detallada de un barco
- **editar_barco.html**: Formulario para editar barcos existentes
- **eliminar_barco.html**: Confirmación de eliminación

> **Nota:** El código completo de cada template está disponible en los archivos del proyecto.

---

11. Fase 9: Migraciones y Base de Datos

11.1 Crear Migraciones

```
# Crear las migraciones para los modelos
uv run python manage.py makemigrations
```

Salida esperada:

```
Migrations for 'core':
  core/migrations/0001_initial.py
    - Create model Usuario
    - Create model Barco
```

11.2 Aplicar Migraciones

```
# Aplicar todas las migraciones a la base de datos
uv run python manage.py migrate
```

Salida esperada:

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, core, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
```

```
  Applying auth.0001_initial... OK
  ...
  Applying core.0001_initial... OK
  Applying sessions.0001_initial... OK
```

11.3 Verificar Estado

```
# Ver el estado de todas las migraciones
uv run python manage.py showmigrations
```

---

12. Fase 10: Datos de Demostración

12.1 Script de Creación de Usuarios

Archivo: `crear_usuarios.py`

```
"""
Script para crear usuarios de prueba en el sistema.
Ejecutar con: uv run python crear_usuarios.py
"""

import os
import sys
import django

# Configurar Django
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'config.settings')
sys.path.insert(0, os.path.dirname(os.path.abspath(__file__)))
django.setup()

from core.models import Usuario, Barco

def crear_usuarios_demo():
    """Crea los usuarios de demostración para probar el sistema."""

    print("=" * 50)
    print("  Creando usuarios de demostración...")
    print("=" * 50)

    usuarios_demo = [
        {
            'username': 'admin',
            'email': 'admin@puerto.com',
            'password': 'admin123',
            'first_name': 'Administrador',
```

```
            'last_name': 'Sistema',
            'rol': 'admin',
            'is_staff': True,
            'is_superuser': True,
        },
        {
            'username': 'ana_operadora',
            'email': 'ana@puerto.com',
            'password': 'operador123',
            'first_name': 'Ana',
            'last_name': 'García',
            'rol': 'operador',
            'is_staff': False,
            'is_superuser': False,
        },
        {
            'username': 'pepe_guardia',
            'email': 'pepe@puerto.com',
            'password': 'guardia123',
            'first_name': 'José',
            'last_name': 'Pérez',
            'rol': 'guardia',
            'is_staff': False,
            'is_superuser': False,
        },
]

for datos in usuarios_demo:
    username = datos.pop('username')
    password = datos.pop('password')

    usuario, creado = Usuario.objects.get_or_create(
        username=username,
        defaults=datos
    )

    if creado:
        usuario.set_password(password)
        usuario.save()
        print(f"  Usuario '{username}' creado con rol '{usuario.rol}'")
    else:
        print(f"   Usuario '{username}' ya existe")

print()
print("=" * 50)
print("  Usuarios disponibles para login:")
```

```
    print("=" * 50)
    print()
    print("                                ")
    print("  Usuario            Contraseña    Rol               ")
    print("                                ")
    print("  admin              admin123      Administrador     ")
    print("  ana_operadora      operador123   Operador          ")
    print("  pepe_guardia       guardia123    Guardia           ")
    print("                                ")
    print()


def crear_barcos_demo():
    """Crea algunos barcos de ejemplo para demostración."""

    print("  Creando barcos de demostración...")
    print()

    try:
        operador = Usuario.objects.get(username='ana_operadora')
    except Usuario.DoesNotExist:
        operador = None

    barcos_demo = [
        {
            'nombre': 'MSC Esperanza',
            'imo': '9484525',
            'bandera': 'Panamá',
            'tipo': 'carga',
        },
        {
            'nombre': 'Costa Pacífica',
            'imo': '9378498',
            'bandera': 'Italia',
            'tipo': 'pasajeros',
        },
        {
            'nombre': 'Tanker Ecuador',
            'imo': '9156778',
            'bandera': 'Ecuador',
            'tipo': 'petrolero',
        },
    ]

    for datos in barcos_demo:
        imo = datos['imo']
```

```
        barco, creado = Barco.objects.get_or_create(
            imo=imo,
            defaults={**datos, 'registrado_por': operador}
        )

        if creado:
            print(f"  Barco '{barco.nombre}' registrado (IMO: {barco.imo})")
        else:
            print(f"   Barco con IMO '{imo}' ya existe")

    print()


if __name__ == '__main__':
    crear_usuarios_demo()
    crear_barcos_demo()
    print("  ¡Datos de demostración creados exitosamente!")
    print()
    print("Ejecuta el servidor con:")
    print("  uv run python manage.py runserver")
    print()
    print("Luego accede a: http://127.0.0.1:8000/")
    print()
```

12.2 Ejecutar el Script

```
uv run python crear_usuarios.py
```

---

13. Pruebas del Sistema

13.1 Iniciar el Servidor

```
uv run python manage.py runserver
```

13.2 Acceder al Sistema

Abrir en el navegador: **http://127.0.0.1:8000/**

13.3 Casos de Prueba

Prueba 1: Usuario Guardia (Solo Lectura)

1. Ingresar como `pepe_guardia` / `guardia123`
2. **Verificar:**
   - Puede ver la lista de barcos
   - Puede ver el detalle de un barco

- NO ve el botón "Registrar Barco" en el menú
- Si intenta acceder a `/barcos/nuevo/` directamente, es redirigido

Prueba 2: Usuario Operador

1. Ingresar como `ana_operadora` / `operador123`
2. **Verificar:**
   - Puede ver la lista de barcos
   - Puede registrar nuevos barcos
   - Puede editar barcos existentes
   - NO puede eliminar barcos

Prueba 3: Usuario Administrador

1. Ingresar como `admin` / `admin123`
2. **Verificar:**
   - Acceso total a todas las funciones
   - Puede eliminar barcos
   - Puede acceder al Panel de Administración (`/admin/`)

---

14. Estructura Final del Proyecto

```
puerto_seguro/
   .git/
   .gitignore
   .python-version
   .venv/                          # Entorno virtual
   config/                         # Configuración Django
       __init__.py
       asgi.py
       settings.py                 #   Configuración principal
       urls.py                     #   Rutas del proyecto
       wsgi.py
   core/                           # Aplicación principal
       __init__.py
       admin.py                    #   Configuración del admin
       apps.py
       forms.py                    #   Formularios
       models.py                   #   Modelos (Usuario, Barco)
       tests.py
       views.py                    #   Vistas con seguridad
       migrations/
           __init__.py
           0001_initial.py
   templates/                      # Plantillas HTML
```

```
    base.html                   #   Template base
    home.html
    dashboard.html
    lista_barcos.html
    crear_barco.html
    detalle_barco.html
    editar_barco.html
    eliminar_barco.html
    registration/
        login.html              #   Página de login
crear_usuarios.py               # Script de datos demo
db.sqlite3                      # Base de datos SQLite
manage.py
pyproject.toml                  # Configuración del proyecto
README.md
uv.lock
```

---

15. Conceptos de Seguridad Implementados

15.1 Autenticación (Authentication)

¿Qué es? Verificar que el usuario es quien dice ser.

Implementación:

- Sistema de login/logout nativo de Django
- Contraseñas hasheadas automáticamente
- Sesiones seguras

```
# Django maneja esto automáticamente con:
path('accounts/', include('django.contrib.auth.urls'))
```

15.2 Autorización (Authorization)

¿Qué es? Determinar qué puede hacer cada usuario autenticado.

Implementación:

```
# Decorador que verifica el rol
@login_required
@user_passes_test(es_operador)
def crear_barco(request):
    ...
```

15.3 Protección CSRF

¿Qué es? Prevenir ataques Cross-Site Request Forgery.

Implementación:

```
<form method="post">
  {% csrf_token %}
  <!-- Token único por sesión -->
  ...
</form>
```

15.4 Auditoría

¿Qué es? Registrar quién hizo qué y cuándo.

Implementación:

```
# En el modelo
registrado_por = models.ForeignKey(Usuario, ...)

# En la vista
barco.registrado_por = request.user
```

15.5 Seguridad en Templates

¿Qué es? Ocultar elementos de la interfaz según permisos.

Implementación:

```
{% if user.rol == 'operador' or user.is_superuser %}
<a href="{% url 'crear_barco' %}">Registrar Barco</a>
{% endif %}
```

---

16. Recomendaciones y Mejoras Futuras

16.1 Seguridad Adicional

- ☐ Implementar autenticación de dos factores (2FA)
- ☐ Agregar límite de intentos de login fallidos
- ☐ Implementar tokens JWT para API REST
- ☐ Agregar registro de actividad (logs)

16.2 Funcionalidades

- ☐ Historial de cambios en barcos
- ☐ Notificaciones por email
- ☐ Exportar reportes a PDF/Excel
- ☐ Búsqueda avanzada con filtros
- ☐ API REST para integración con otros sistemas

16.3 Interfaz

- ☐ Modo oscuro
- ☐ Internacionalización (múltiples idiomas)
- ☐ Dashboard con gráficos (Chart.js)
- ☐ Diseño responsive mejorado

16.4 Infraestructura

- ☐ Migrar a PostgreSQL para producción
- ☐ Configurar servidor HTTPS
- ☐ Implementar Docker para despliegue
- ☐ Configurar CI/CD

Soporte

Para dudas o sugerencias sobre este proyecto, contactar al equipo de Vinculación ULEAM.
