**Manual Técnico - Autolavados Plataforma**

*Fecha: 2025-10-10 14:32 | Entorno: Producción (PythonAnywhere)*

# 1. Resumen del Sistema

Sistema web para la gestión integral de un autolavado: autenticación de usuarios, gestión de clientes y vehículos, reservas de servicios, notificaciones, paneles de control y administración. Basado en Django 4.2 y MySQL, expone APIs REST para consumo desde frontend y soporta flujos internos mediante vistas clásicas.

# 2. Arquitectura y Tecnologías

* Framework: Django 4.2.x
* Base de datos: MySQL (Producción) / SQLite (Desarrollo)
* API: Django REST Framework (autenticación, permisos, paginación)
* Tiempo real / eventos: Channels (configurado en el proyecto)
* Servidor de aplicación: Gunicorn (producción)
* Archivos estáticos: WhiteNoise (compresión y manifest)
* Almacenamiento de medios: Local por defecto; S3 opcional con django-storages/boto3
* Cache: LocMem por defecto; Redis opcional si REDIS\_URL está definido
* Dependencias clave: Django, asgiref, channels, DRF, cors-headers, mysqlclient, pillow, requests, sqlparse

# 3. Estructura de Aplicaciones

* autenticacion: Modelo de usuario personalizado, formularios y vistas de login/logout
* clientes: Administración de clientes y vehículos
* reservas: Core del sistema (Reservas, Servicios, Bahías, Horarios)
* notificaciones: Envío y gestión de notificaciones a clientes y personal
* empleados: Gestión de personal, incentivos, reportes y utilidades
* dashboards: Vistas públicas y gerenciales para métricas e indicadores

## 3.1 Estructura completa del Proyecto

Ver en GitHub

# 4. Modelos principales (visión general)

* Usuario (autenticacion.Usuario): Extiende el modelo de Django para roles y atributos del sistema
* Cliente y Vehiculo (clientes): Relación y administración de la flota de clientes
* Reserva (reservas): Entidad clave con estado, servicio, medio de pago y asignaciones
* Servicio y Bahia (reservas): Catálogo de servicios y espacios físicos de atención
* Notificacion (notificaciones): Mensajes informativos, estados y tipos

# 5. Middleware personalizado

* CSRFDebugMiddleware: Diagnóstico de tokens CSRF en flujos AJAX
* AJAXExceptionMiddleware: Manejo centralizado de excepciones para solicitudes AJAX
* LoginRequiredMiddleware: Redirección a login para vistas protegidas
* TimezoneMiddleware: Gestión de zona horaria del usuario

# 6. Configuración de Producción (PythonAnywhere)

* Archivo: settings\_production.py
* Dominio PythonAnywhere: <usuario>.pythonanywhere.com
* SITE\_URL: http://localhost:8000
* ALLOWED\_HOSTS: localhost,127.0.0.1
* Variables de entorno: SECRET\_KEY, DB\_NAME/USER/PASSWORD/HOST/PORT, EMAIL\_\*, DEBUG=false
* Estáticos: STATIC\_ROOT y WhiteNoise para servir assets
* Email: SMTP por defecto (configurable)

# 7. API y Ruteo

* URLs principales en autolavados\_plataforma.urls
* APIs REST: vistas y endpoints en apps (ej. reservas/urls\_api.py, autenticacion/urls.py)
* Autenticación API: Token, Basic y Session (REST\_FRAMEWORK)

## 7.1 Endpoints de Autenticación

* http://localhost:8000/api/auth/api/registro/ [POST] - Registro de nuevos usuarios
* http://localhost:8000/api/auth/api/login/ [POST] - Inicio de sesión
* http://localhost:8000/api/auth/api/logout/ [POST] - Cierre de sesión
* http://localhost:8000/api/auth/api/perfil/ [GET/PUT] - Ver/actualizar perfil

## 7.2 Endpoints de Reservas

* http://localhost:8000/api/reservas/servicios/ [GET/POST] - CRUD de servicios
* http://localhost:8000/api/reservas/reservas/ [GET/POST] - CRUD de reservas
* http://localhost:8000/api/reservas/bahias/ [GET] - Listado de bahías
* http://localhost:8000/api/reservas/verificar-placa/ [POST] - Validar placa única

## 7.3 Endpoints de Notificaciones

* http://localhost:8000/notificaciones/api/contador/ [GET] - Conteo de notificaciones
* http://localhost:8000/notificaciones/api/dropdown/ [GET] - Dropdown de notificaciones
* http://localhost:8000/notificaciones/api/marcar-leida/<id>/ [POST] - Marcar leída
* http://localhost:8000/notificaciones/api/marcar-todas-leidas/ [POST] - Marcar todas leídas

# 8. Dependencias y utilidades

* requirements.txt: conjunto mínimo para producción
* Utilidades: qrcode (códigos QR), reportlab (PDF), openpyxl/xlsxwriter (Excel)
* Opcional S3: django-storages + boto3, si se usa almacenamiento en AWS

# 9. Migraciones y Base de Datos

Las carpetas de migraciones están incluidas en el repositorio para garantizar que el entorno de producción pueda aplicar los cambios de esquema correctamente. Flujo recomendado: makemigrations (desarrollo), commit/push, migrate en producción.

# 10. Seguridad

* SECRET\_KEY se gestiona por variables de entorno y no debe versionarse
* DEBUG debe estar en False en producción
* CSRF y autenticación: protección para formularios y APIs, sesiones seguras

# 11. Pruebas y utilidades de verificación

* Suite de tests localizada en archivos test\_\*.py (API, reservas, empleados, UI, despliegue)
* Scripts de verificación: check\_reservas.py, check\_notifications.py, verificación de conexiones y entorno

# 12. Despliegue en PythonAnywhere (resumen)

* Actualizar código: git pull origin main
* Instalar dependencias: pip install --no-cache-dir -r requirements.txt
* Migraciones: python manage.py migrate --noinput
* Estáticos: python manage.py collectstatic --noinput
* Recargar la web app desde el panel de PythonAnywhere

# 13. Mantenimiento y buenas prácticas

* Separar dependencias de desarrollo si es necesario (requirements-dev.txt)
* Control de cuotas: evitar paquetes pesados innecesarios en producción
* Revisar ALLOWED\_HOSTS y variables de entorno en cada despliegue

# 14. Diagramas de Arquitectura y Flujo

* Arquitectura: ver docs/diagrams/arquitectura.svg
* Flujo de reservas: ver docs/diagrams/flujo\_reservas.svg