



unl

Universidad
Nacional
de Loja

FEIRNNR - Carrera de Computación

10-2-2026

Informe Técnico — Sistema de Control de Acceso Militar

INTEGRACIÓN MYSQL ↔
MONGODB, APIS EN DJANGO Y
PANEL DE ANALÍTICAS

Wilson Palma

UNIVERSIDAD NACIONAL DE LOJA



INFORME TÉCNICO — SISTEMA DE CONTROL DE ACCESO MILITAR

INTEGRACIÓN MYSQL ↔ MONGODB, APIS EN DJANGO Y PANEL DE ANALÍTICAS

WILSON PALMA

2026-02-10

Resumen

Este informe documenta el diseño, implementación y despliegue del sistema **Control de Acceso Militar**, una solución que integra una base de datos relacional (MySQL) para datos maestros y una base de datos NoSQL (MongoDB) para el registro histórico de intentos de acceso. Se describen el esquema de datos, los servicios web (WS) desarrollados en Django para validación y exposición de recursos, la web-application (WA) que consume esos servicios, la generación de datos de prueba y el módulo de analíticas con visualizaciones interactivas. El documento incluye instrucciones de inicialización, ejemplos de uso y recomendaciones operativas.

1 CONCEPTUALIZACIÓN Y DISEÑO DEL SISTEMA

El proyecto nace de la necesidad de gestionar el acceso de personal militar a zonas restringidas mediante un sistema de validación jerárquica. Se decidió una arquitectura de dos componentes:

- **Web Service (WS):** Encargado de la lógica de negocio y gestión de datos maestros en **MySQL**.
- **Web Application (WA):** Interfaz de usuario para el registro de intentos y almacenamiento de logs en **MongoDB**.

2 INFRAESTRUCTURA DE DATOS

2.1 BASE DE DATOS RELACIONAL (MYSQL: SISTEMA_ACCESO_MILITAR)

Se diseñó un esquema relacional para gestionar la jerarquía y permisos del personal. Las tablas principales incluyen:

- **Personal y Rangos:** `personnel`, `ranks` (con niveles numéricos para comparaciones) y `units`.
- **Seguridad:** `clearance_levels` y `permissions` especiales.
- **Zonas:** `restricted_zones` que definen requisitos de rango y autorización.

Recurso Externo: [Link al archivo esquema+tuplas.sql en Drive](#)

Nota: Este script incluye la creación de tablas y procedimientos almacenados para el sembrado (seeding) aleatorio de personal y permisos.

2.2 BASE DE DATOS NOSQL (MONGODB: HISTORIAL_REGISTROS_ACCESO_MILITAR)

Para los registros de acceso, se optó por un modelo **denormalizado** en la colección `access_attempts`. Cada documento es un “snapshot” inmutable que contiene toda la información del militar (rango, unidad, clearance) y de la zona en el momento exacto del intento, evitando la necesidad de joins complejos en las analíticas.

3 DESARROLLO DEL WEB SERVICE (WS) — MYSQL

El objetivo de este componente es exponer una **API REST** para que otros sistemas consulten la elegibilidad de un militar.

- **Modelado:** Sincronización de los modelos de Django con las tablas de MySQL existentes.
- **Endpoints Clave:**
 - Exposición de zonas disponibles para el formulario del WA.
 - Validación de acceso (`access-info`): Recibe el ID del badge o número de servicio y el código de zona para devolver un veredicto basado en reglas de negocio.

4 DESARROLLO DE LA WEB APPLICATION (WA) — MONGODB

Esta aplicación gestiona la interacción del guardia o sistema de control en los puntos de acceso.

- **Formulario de Registro:** Permite ingresar el `service_number` o `badge_id` y seleccionar una zona (obtenida dinámicamente desde el WS).
- **Lógica de Integración:** Al enviar el formulario, el WA consulta al WS. Si el militar existe, procesa la respuesta y genera un documento en MongoDB con el resultado (`allowed: true/false`) y la evidencia de la validación.
- **Estructura de Archivos:** Se implementaron módulos específicos como `mongo.py` para la conexión a la base NoSQL y `services.py` para la lógica de consumo de APIs.

5 MÓDULO DE ANALÍTICAS Y DASHBOARD

Una vez recolectados los datos en MongoDB, se desarrolló un panel de control para la visualización táctica de los datos.

- **Procesamiento:** Uso de **Pandas** y **Numpy** para procesar los logs de acceso.
- **Visualización:** Implementación de gráficos (barras, pastel y otros comunes) utilizando **Chart.js** para representar métricas como:
 - Intentos de acceso por zona.
 - Relación de accesos permitidos vs. denegados.
 - “Top offenders” (personal con más intentos denegados).



- **Mapeo de Datos:** El dashboard incluye tablas de correspondencia debajo de las gráficas para facilitar la lectura de códigos de zonas y rangos.

6 INSTRUCCIONES DE INICIALIZACIÓN

Para replicar el entorno de desarrollo, se deben seguir estos pasos documentados:

1. **Carga de DBs:** Ejecutar el [script SQL](#) en MySQL y restaurar el [Dump](#) de MongoDB.
2. **Entorno Python:** Crear un entorno virtual e instalar dependencias (incluyendo `pymysql` y `dnspython`).
3. **Configuración:** Ajustar el archivo `settings.py` con las credenciales de ambas bases de datos.
4. **Ejecución:** Realizar migraciones de Django y lanzar el servidor para ambos proyectos.

Este proceso asegura un sistema desacoplado donde los cambios en los datos maestros de MySQL no afectan la integridad histórica de los registros en MongoDB.