**4.6 SEGURIDAD**

Este capítulo describe las medidas de seguridad implementadas en el sistema **Programa de Alimentación Escolar (PAE)** desarrollado con **Python Flask y MySQL**, siguiendo las mejores prácticas de seguridad web y normativas aplicables.

**🔹 1. Auditoría de Seguridad y Capacidad**

✅ **Pruebas de Estrés**: Se realizarán pruebas de carga y estrés para evaluar el rendimiento del sistema bajo condiciones extremas.  
✅ **Pruebas de Seguridad (OWASP Top 10)**: Se auditarán vulnerabilidades como **inyección SQL, XSS, CSRF, autenticación rota**, entre otras.  
✅ **Entrega de Resultados**: Se generará un informe de auditoría detallado antes de pasar a producción.

**🔹 2. Registro de Transacciones y Auditoría**

✅ **Todas las operaciones quedan registradas** en la base de datos para auditoría. Esto incluye:

* **Fecha y hora** de la acción.
* **Módulo y contenido** afectado.
* **Tipo de acción**: Creación, modificación, eliminación, consulta.
* **Usuario y rol** que realizó la operación.
* **IP de origen** de la operación.  
  ✅ **Implementación**: Se usa **SQL con triggers y logs** para garantizar la trazabilidad de cada operación.

**🔹 3. Gestión de Roles y Permisos**

✅ Se ha implementado un sistema de **roles y permisos en MySQL**, con restricciones a nivel de aplicación y base de datos.  
✅ **Roles definidos** en el sistema:

* **Administrador**: Acceso total.
* **Supervisor**: Auditoría y reportes.
* **Operador**: Gestión de intercambios y solicitudes.
* **Nutricionista**: Evaluación de menús e intercambios.  
  ✅ Control de accesos basado en **JWT y sesiones seguras en Flask**.

**🔹 4. Desarrollo Seguro (Buenas Prácticas)**

✅ **Metodología de Desarrollo Seguro**: Se siguen prácticas recomendadas por **OWASP y Flask Security**.  
✅ **Protección contra ataques comunes**:

* **Inyección SQL**: Uso de consultas parametrizadas.
* **Cross-Site Scripting (XSS)**: Filtrado y escape de datos en plantillas HTML.
* **Cross-Site Request Forgery (CSRF)**: Uso de tokens CSRF en formularios.
* **Seguridad en contraseñas**: Hash con **bcrypt** antes de almacenamiento.

**🔹 5. Infraestructura y Requerimientos Técnicos**

✅ **Sistema Operativo**: Se recomienda **Linux (Ubuntu Server 20.04 o superior)**.  
✅ **Servidor Web**: Se usa **Gunicorn con Nginx** para balanceo de carga.  
✅ **Base de Datos**: MySQL con **usuarios restringidos sin acceso root**.  
✅ **Librerías Python**: Documentadas en requirements.txt, permitiendo control sobre versiones y actualizaciones.

**🔹 6. Permisos en el Servidor y Archivos**

✅ **Restricciones en usuarios del sistema operativo**:

* No se permite acceso root para usuarios de aplicación.
* Se definen permisos mínimos necesarios para cada usuario.  
  ✅ **Permisos sobre archivos y directorios**:
* Archivos críticos en modo **read-only**.
* Solo los administradores pueden modificar configuraciones sensibles.  
  ✅ **Límites en subida de archivos**:
* Tamaño máximo definido.
* Restricción de extensiones para evitar archivos maliciosos.
* Filtrado de nombres para evitar inyecciones.

**🔹 7. Seguridad en la Administración y Comunicaciones**

✅ **Acceso administrativo restringido a la red interna**.  
✅ **Cifrado HTTPS con TLS 1.2+** para todas las comunicaciones.  
✅ **Separación de capas en la arquitectura**:

* **Frontend** (HTML + JavaScript).
* **Backend (Flask en Python)**.
* **Base de Datos (MySQL en servidor separado)**.  
  ✅ **No se permite modificar parámetros por URL** para evitar manipulación indebida.