Guía del Estudiante - Video 3

Diagnóstico y Validación de LogInsights

Código utilizado en el video

1. Objetivo

Implementar scripts de diagnóstico y validación para verificar el correcto funcionamiento del sistema LogInsights, incluyendo la generación de reportes, conectividad entre servicios y análisis de la calidad de los análisis de IA.

2. Estructura del proyecto

```
proyecto/
          docker-compose.yml
          .env
          loginsights-docker/
                Dockerfile
                loginsights.py
                entrypoint.sh
          diagnostico.sh
          validacion.sh
```

3. Script de Diagnóstico (diagnostico.sh)

```
#!/usr/bin/env bash
echo "=== DIAGN STICO DE LOGINSIGHTS ==="
echo
echo "1. Contenedores y su estado:"
docker ps -a --format "table {{.Names}}\t{{.Status}}\t{{.State}}" | grep -E "(moodle|ollama|loginsights|db)"
echo
echo "2. Verificando volumen de reportes:"
docker exec loginsights ls -la /reports 2>/dev/null || echo "No se puede acceder a /reports"
echo
echo "3. Reportes generados:"
ls -la ./log_reports/ 2>/dev/null || echo "No hay reportes en ./log_reports/"
echo
echo "4. Verificando conexi n a Ollama:"
docker exec loginsights curl -s http://ollama:11434/api/tags | jq '.models[].name' 2>/dev/null || echo "No se puede conectar a Ollama"
echo
```

```
echo "5.  ltimos  logs de loginsights (30 l neas):"
docker logs loginsights --tail 30 2>&1
echo
echo "6. Estado de la base de datos:"
docker logs moodle-db --tail 20 2>&1 | grep -E "(ready|error|started)" || echo "No hay logs relevantes"
echo
echo "7. Verificando permisos del socket Docker:"
docker exec loginsights ls -la /var/run/docker.sock 2>/dev/null || echo "No se puede acceder al socket"
echo
echo "8. Verificando script de LogInsights:"
docker exec loginsights python3 -c "import loginsights; print('LogInsights m dulo cargado correctamente')" 2>/dev/null || echo "No se puede verificar el script"
echo
echo "9. Probando an lisis con Ollama directamente:"
docker exec loginsights curl -s -X POST http://ollama:11434/api/generate \
  -H "Content-Type: application/json" \
  -d '{"model":"tinyllama:1.1b","prompt":"Test: Say hello","stream":false}' | jq -r '.response // "No response"' 2>/dev/null || echo "No se puede conectar a Ollama"
echo
echo "10. Mostrando un reporte reciente completo:"
LATEST_REPORT=$(docker exec loginsights find /reports -name "summary_*.txt" -type f -printf '%T@ %p\n' 2>/dev/null | sort -n | tail -1 | awk '{print $2}')
if [ -n "$LATEST_REPORT" ]; then
    echo "Reporte m s reciente: $(basename $LATEST_REPORT)"
    echo "---"
    docker exec loginsights head -50 "$LATEST_REPORT" 2>/dev/null
    echo "---"
else
    echo "No hay reportes disponibles"
fi
```

Listing 1: Script de diagnóstico completo

4. Script de Validación (validacion.sh)

```
#!/usr/bin/env bash
# Colors for output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
NC='\033[0m' # No Color

echo -e "${BLUE}=== LOGINSIGHTS VALIDATION SCRIPT ===${NC}"
echo "Timestamp: $(date)"
echo

# Variables
TOTAL_TESTS=0
PASSED_TESTS=0
WARNINGS=0

# Helper functions
check_pass() {
    TOTAL_TESTS=$((TOTAL_TESTS + 1))
    PASSED_TESTS=$((PASSED_TESTS + 1))
    echo -e "${GREEN}    $1${NC}"
}

check_fail() {
    TOTAL_TESTS=$((TOTAL_TESTS + 1))
    echo -e "${RED}    $1${NC}"
}

check_warn() {
    WARNINGS=$((WARNINGS + 1))
    echo -e "${YELLOW}    $1${NC}"
}

# 1. Check container status
echo -e "${BLUE}1. CHECKING CONTAINER STATUS${NC}"
for container in loginsights moodle-app moodle-db ollama; do
    if docker ps --format "{{.Names}}" | grep -q "^${container}$"; then
        check_pass "$container is running"
    else
        check_fail "$container is NOT running"
    fi
done
echo

# 2. Check report generation
echo -e "${BLUE}2. CHECKING REPORT GENERATION${NC}"
REPORT_COUNT=$(docker exec loginsights find /reports -name "summary_*.
   txt" -type f 2>/dev/null | wc -l)
if [ "$REPORT_COUNT" -gt 0 ]; then
    check_pass "Found $REPORT_COUNT reports"
else
    check_fail "No reports found"
fi

# Check if reports are being generated for all containers
```

```
for container in moodle-app moodle-db ollama; do
    CONTAINER_REPORTS=$(docker exec loginsights find /reports -name "summary_${container}_*.txt" -type f 2>/dev/null | wc -l)
    if [ "$CONTAINER_REPORTS" -gt 0 ]; then
        check_pass "Found $CONTAINER_REPORTS reports for $container"
    else
        check_fail "No reports found for $container"
    fi
done
echo

# 3. Check report freshness
echo -e "${BLUE}3. CHECKING REPORT FRESHNESS${NC}"
LATEST_REPORT=$(docker exec loginsights find /reports -name "summary_*.txt" -type f -printf '%T@ %p\n' 2>/dev/null | sort -n | tail -1 | awk '{print $2}')
if [ -n "$LATEST_REPORT" ]; then
    REPORT_AGE=$(docker exec loginsights bash -c "echo \$((\$(date +%s) - \$(stat -c %Y '$LATEST_REPORT')))")
    if [ "$REPORT_AGE" -lt 180 ]; then
        check_pass "Latest report is $REPORT_AGE seconds old (fresh)"
    elif [ "$REPORT_AGE" -lt 300 ]; then
        check_warn "Latest report is $REPORT_AGE seconds old (slightly old)"
    else
        check_fail "Latest report is $REPORT_AGE seconds old (stale)"
    fi
fi
echo
```

Listing 2: Script de validación con colores y métricas

```
# 4. Check report content quality
echo -e "${BLUE}4. CHECKING REPORT CONTENT QUALITY${NC}"
# Get the most recent report for each container
for container in moodle-app moodle-db ollama; do
    LATEST=$(docker exec loginsights find /reports -name "summary_${container}_*.txt" -type f 2>/dev/null | sort | tail -1)
    if [ -n "$LATEST" ]; then
        echo -e "\n${YELLOW}Checking $container report: $(basename $LATEST)${NC}"

        # Check file size
        SIZE=$(docker exec loginsights stat -c%s "$LATEST" 2>/dev/null || echo 0)
        if [ "$SIZE" -gt 1000 ]; then
            check_pass "Report size is adequate ($SIZE bytes)"
        else
            check_warn "Report might be too small ($SIZE bytes)"
        fi

        # Check for key sections
        if docker exec loginsights grep -q "=== AN LISIS ===" "$LATEST" 2>/dev/null; then
            check_pass "Contains analysis section"
        else
            check_fail "Missing analysis section"
        fi
```

```
23
24          if docker exec loginsights grep -q "=== LOGS ORIGINALES" "
               $LATEST" 2>/dev/null; then
25              check_pass "Contains original logs section"
26          else
27              check_fail "Missing original logs section"
28          fi
29
30          # Check if analysis was successful or timed out
31          if docker exec loginsights grep -q "timeout" "$LATEST" 2>/dev/
               null; then
32              check_warn "Report indicates timeout occurred"
33          fi
34
35          # Check for actual log content
36          LOG_LINES=$(docker exec loginsights grep -E "^[0-9]{4}-|^[A-Za-z
               ]{3} [0-9]{2}" "$LATEST" 2>/dev/null | wc -l || echo 0)
37          if [ "$LOG_LINES" -gt 10 ]; then
38              check_pass "Contains $LOG_LINES log lines"
39          elif [ "$LOG_LINES" -gt 0 ]; then
40              check_warn "Only contains $LOG_LINES log lines"
41          else
42              check_fail "No log lines found"
43          fi
44      fi
45  done
46  echo
47
48  # 5. Check Ollama connectivity
49  echo -e "${BLUE}5. CHECKING OLLAMA CONNECTIVITY${NC}"
50  if docker exec loginsights curl -s http://ollama:11434/api/tags >/dev/
       null 2>&1; then
51      check_pass "Ollama API is accessible"
52
53      MODEL_CHECK=$(docker exec loginsights curl -s http://ollama:11434/
           api/tags | grep -o 'tinyllama:1.1b' || echo "")
54      if [ -n "$MODEL_CHECK" ]; then
55          check_pass "tinyllama:1.1b model is available"
56      else
57          check_fail "tinyllama:1.1b model not found"
58      fi
59  else
60      check_fail "Cannot connect to Ollama API"
61  fi
62  echo
63
64  # 6. Check Docker socket permissions
65  echo -e "${BLUE}6. CHECKING DOCKER PERMISSIONS${NC}"
66  SOCKET_PERMS=$(docker exec loginsights stat -c "%a" /var/run/docker.sock
       2>/dev/null || echo "")
67  if [ -n "$SOCKET_PERMS" ]; then
68      check_pass "Docker socket is accessible"
69      if [ "$SOCKET_PERMS" = "660" ] || [ "$SOCKET_PERMS" = "666" ]; then
70          check_pass "Docker socket has correct permissions ($SOCKET_PERMS
               )"
71      else
72          check_warn "Docker socket has unusual permissions ($SOCKET_PERMS
               )"
```

```
    fi
else
    check_fail "Cannot access Docker socket"
fi
echo

# 7. Check for errors in LogInsights logs
echo -e "${BLUE}7. CHECKING FOR ERRORS${NC}"
ERROR_COUNT=$(docker logs loginsights 2>&1 | grep -iE "(error|exception|traceback)" | grep -v "NewConnectionError" | wc -l)
if [ "$ERROR_COUNT" -eq 0 ]; then
    check_pass "No errors found in LogInsights logs"
else
    check_warn "Found $ERROR_COUNT error messages in logs"
    echo "Recent errors:"
    docker logs loginsights 2>&1 | grep -iE "(error|exception|traceback)" | grep -v "NewConnectionError" | tail -5
fi
echo

# 8. Performance check
echo -e "${BLUE}8. CHECKING PERFORMANCE${NC}"
# Check memory usage
MEMORY_USAGE=$(docker stats --no-stream --format "{{.MemUsage}}" loginsights | awk '{print $1}')
echo "Memory usage: $MEMORY_USAGE"

# Check if reports are generated on schedule
REPORT_TIMES=$(docker exec loginsights find /reports -name "summary_*.txt" -type f -printf '%TY%Tm%Td_%TH%TM%TS\n' 2>/dev/null | sort | tail -5)
if [ -n "$REPORT_TIMES" ]; then
    check_pass "Recent report generation times:"
    echo "$REPORT_TIMES" | while read -r time; do
        echo "  - $time"
    done
fi
echo

# 9. Display sample analysis
echo -e "${BLUE}9. SAMPLE ANALYSIS OUTPUT${NC}"
SAMPLE_REPORT=$(docker exec loginsights find /reports -name "summary_*.txt" -type f 2>/dev/null | sort | tail -1)
if [ -n "$SAMPLE_REPORT" ]; then
    echo -e "${YELLOW}From: $(basename $SAMPLE_REPORT)${NC}"
    echo "---"
    docker exec loginsights bash -c "sed -n '/=== AN LISIS ===/,/=== LOGS ORIGINALES/{/=== LOGS ORIGINALES/!p}' '$SAMPLE_REPORT' | head -20" 2>/dev/null || echo "Could not extract analysis"
    echo "---"
fi
echo

# Final summary
echo -e "${BLUE}=== VALIDATION SUMMARY ===${NC}"
echo -e "Total tests: $TOTAL_TESTS"
echo -e "Passed: ${GREEN}$PASSED_TESTS${NC}"
echo -e "Failed: ${RED}$((TOTAL_TESTS - PASSED_TESTS))${NC}"
```

```
123 echo -e "Warnings: ${YELLOW}$WARNINGS${NC}"
```

Listing 3: Continuación del script de validación

5. Comandos de Configuración y Ejecución

5.1. Configuración inicial

```
1 # Generar clave SSH para autenticaci n
2 ssh-keygen -t ed25519 -C "tu_correo@ejemplo.com"
```

Listing 4: Generación de claves SSH

5.2. Comandos de Docker

```
1 # Verificar estado de contenedores
2 docker ps
3
4 # Levantar toda la infraestructura
5 docker compose up -d
6
7 # Dar permisos de ejecuci n a los scripts
8 chmod +x ./diagnostico.sh
9 chmod +x ./validacion.sh
```

Listing 5: Comandos básicos de gestión

5.3. Ejecución de los scripts

```
1 # Ejecutar diagn stico general
2 ./diagnostico.sh
3
4 # Ejecutar validaci n completa
5 ./validacion.sh
6
7 # Ejecutar diagn stico con redirecci n de salida
8 ./diagnostico.sh > diagnostico_$(date +%Y%m%d_%H%M%S).log
9
10 # Ejecutar validaci n y guardar resultados
11 ./validacion.sh | tee validacion_$(date +%Y%m%d_%H%M%S).log
```

Listing 6: Comandos de diagnóstico y validación