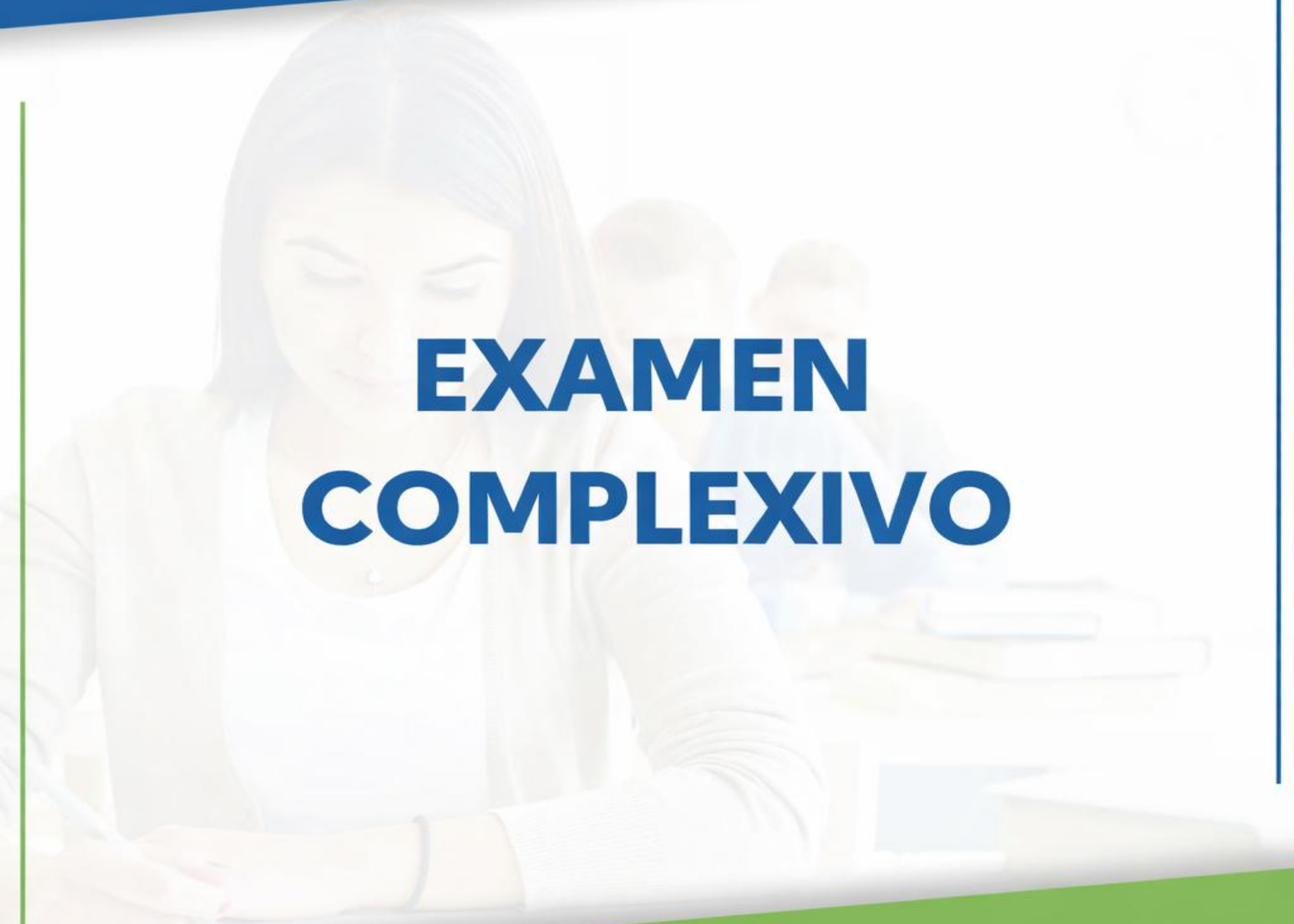
UNIDAD DE INTEGRACIÓN CURRICULAR

CARRERA: TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE



EXAMEN COMPLEXIVO

UTE ESCUELA DE TECNOLOGÍAS

PERÍODO ACADÉMICO

2025-II

1. EXAMEN PRÁCTICO: Sistema Básico de Gestión de Órdenes y Operaciones para un Laboratorio Clínico

1. t ítulo del caso:	**Sistema Básico de Gestión de Órdenes y Operaciones para un Laboratorio Clínico**
2. N2 Áreas de la carrera:	**1.** Tecnologías de la Información **2.** Diseño y Desarrollo de Software
3. N3 Subáreas de la carrera relacionadas con el perfil de egreso:	1. Análisis de Datos y Sistemas 2. Base de Datos 3. Programación 4. Calidad y Gestión
8. N4 Materias:	1. Base de Datos I 2. Base de Datos II 3. Programación III 4. Programación IV 5. Sistemas Operativos
9. Identificación de resultados de aprendizaje relacionados con el caso:	El estudiante demuestra que es capaz de: • Diseñar y administrar bases de datos SQL y NoSQL. • Implementar un backend con API REST. • Consumir servicios desde aplicaciones web y móviles. • Aplicar principios básicos de seguridad y validación. • Ejecutar pruebas unitarias. • Configurar y ejecutar un sistema en Linux. • Integrartodas las capas del sistema de forma funcional.
10. Objetivo	Desarrollar, documentar y presentar una solución de software completa que permita gestionar servicios internos de una organización, integrandotodas las capas del stacktecnológico (front-end, back-end, bases de datos SQL y NoSQL, seguridad, pruebas y despliegue en Linux), mediante metodologías de gestión de proyectos y herramientas colaborativas.

11. Planteamiento:	Diseñe e implemente un **Sistema Básico de Gestión de Órdenes y Operaciones para un Laboratorio Clínico**, integrando una base de datos relacional, una base de datos no relacional, un backend con API, una interfaz web, una aplicación móvil y su ejecución en un entorno Linux. El sistema debe permitir registrar órdenes de producción, consultar su estado y almacenar eventos operativos, demostrando la integración completa del stack tecnológico y la aplicación de buenas prácticas de desarrollo.
12. Descripción de entregables	**Back-end funcional** • API REST u otra arquitectura definida • Validaciones, control de acceso, sanitización **Front-end funcional** • Interfaz implementada • Integración con API • Manejo de estados y errores **Proyecto Móvil funcional** • Interfaz implementada • Integración con API • Manejo de estados y errores **Base de datos relacional (SQL)** • Scripts de creación • Consultas necesarias **Base de datos NoSQL** • Colecciones, documentos • Consultas **Integración completa del sistema** • Front-end ↔ API • Proyecto Móvil↔ API • API ↔ SQL y NoSQL • Flujo end-to-end funcionando **Ejecución en Linux** • Comandos utilizados • Configuración necesaria • Evidencia de funcionamiento enterminal

ENLACE GITHUB: https://github.com/ElianMuriel/Laboratorio_complexivo_ute.git

CASO: Sistema Básico de Gestión de Órdenes y Operaciones para un Laboratorio Clínico

Regla de negocio: gestionar pruebas de laboratorio y órdenes clínicas con estados (CREATED, PROCESSING, COMPLETED, CANCELLED), y registrar operación NoSQL (catálogo/bitácora y eventos) vinculada a cada orden. Misma estructura técnica: 2 tablas (PostgreSQL) + 2 colecciones (MongoDB), ReactJS consume tablas, React Native consume colecciones, integración por lab_order_id (id SQL) en eventos NoSQL.

MODELO DE DATOS OBLIGATORIO (2 tablas + 2 colecciones)

Tablas PostgreSQL (2):

Tabla lab_tests (pruebas):

id BIGSERIAL PRIMARY KEY
test_name VARCHAR(150) NOT NULL
sample_type VARCHAR(60) NOT NULL
price NUMERIC(10,2) NOT NULL
is_available BOOLEAN NOT NULL DEFAULT TRUE

Tabla lab_orders (órdenes):

id BIGSERIAL PRIMARY KEY
test_id BIGINT NOT NULL REFERENCES lab_tests(id)
patient_name VARCHAR(120) NOT NULL
status VARCHAR(20) NOT NULL (CREATED, PROCESSING, COMPLETED, CANCELLED)
result_summary VARCHAR(300) NULL
created_at TIMESTAMP NOT NULL DEFAULT NOW()

Colecciones MongoDB (2):

Colección test_catalog (catálogo / referencia):
_id ObjectId
test_name string
category string
normal_range string
method string
is_active bool

Colección lab_order_events (eventos operativos):
_id ObjectId
lab_order_id long
event_type string (CREATED, PROCESSING, COMPLETED, CANCELLED, RESULT_UPDATED)
source string (WEB, MOBILE, SYSTEM)
note string
created_at date

Nota de integración: lab_order_events.lab_order_id debe guardar el id de la orden creada en PostgreSQL.

SECUENCIA LÓGICA (PASO A PASO, APOYÁNDOSE EN MIGRACIONES DJANGO)

1. Crear BD y usuario en PostgreSQL
2. Crear proyecto Django y modelos
3. Migrar (Django crea tablas)
4. Verificar tablas en psql
5. Crear endpoints SQL (lab-tests, lab-orders)
6. Crear DB y usuario en Mongo
7. Crear colecciones e índices
8. Crear endpoints Mongo (test-catalog, lab-order-events)
9. Integrar: al crear lab_order en SQL generar evento en Mongo
10. ReactJS consume SQL (lab_tests y lab_orders)
11. Mobile consume Mongo (test_catalog y lab_order_events)
12. Evidencia en Linux + Git

BASE DE DATOS I – PostgreSQL (8 preguntas)

Objetivo: Crear y configurar una base de datos relacional en PostgreSQL para el backend, asegurando gestión de usuarios, permisos, integridad y optimización.

1. Crear la base de datos lab_db en PostgreSQL desde terminal/psql.
2. Crear el usuario backend_user con contraseña segura y sin privilegios de superusuario.
3. Asignar permisos mínimos para migraciones Django (conectar, usar schema, crear/alterar objetos).
4. Verificar desde psql conexión con backend_user y existencia de lab_db.
5. Ejecutar migraciones Django y verificar que existen las tablas lab_tests y lab_orders (listar tablas y un select).
6. Crear un índice en lab_orders(status) y demostrar su uso con consultas (select * from lab_orders; select * from lab_tests;).
7. Crear una vista vw_active_lab_orders que liste únicamente órdenes en estado CREATED o PROCESSING.
8. Crear una función o trigger (uno): función que retorne total de órdenes por estado o trigger que impida price <= 0 o test_id inválido a nivel DB (además de validación en

backend).

BASE DE DATOS II – MongoDB (7 preguntas)

Objetivo: Diseñar e implementar una base de datos NoSQL para eventos y registros del sistema.

1. Crear/definir la base de datos lab_logs (evidenciar use lab_logs en mongosh).
2. Crear el usuario mongo_backend_user con contraseña exa_2026_ute.
3. Asignar roles mínimos para leer/escribir en lab_logs (sin permisos administrativos globales).
4. Probar autenticación conectándose con mongo_backend_user y verificar que puede operar sobre lab_logs.
5. Verificar/crear colecciones test_catalog y lab_order_events con los campos definidos.
6. Crear índice en lab_order_events(lab_order_id) y evidenciar con getIndexes().
7. Ejecutar 2 consultas: eventos por lab_order_id y eventos por rango de fechas (created_at).

PROGRAMACIÓN III – Backend Django + Frontend React (9 preguntas)

Backend – Django REST

1. Crear proyecto Django lab_backend.
2. Crear app lab.
3. Configurar conexión a PostgreSQL (lab_db).
4. Configurar conexión a MongoDB (lab_logs).
5. Crear modelos Django para lab_tests y lab_orders según campos definidos.
6. Crear endpoint GET /lab-tests que liste pruebas (SQL).
7. Crear endpoints GET /lab-orders y POST /lab-orders (SQL), y en POST registrar evento en lab_order_events (Mongo) con lab_order_id.

Frontend – ReactJS (consume TABLAS SQL)

8. Crear proyecto React.
9. Crear vistas que consuman GET /lab-tests y GET /lab-orders y muestren pruebas y órdenes con estado, manejo de carga/error y actualización tras crear orden.

PROGRAMACIÓN IV – React Native (9 preguntas)

Objetivo: Consumir la API desde una app móvil básica enfocada a NoSQL.

1. Crear proyecto React Native.

2. Configurar conexión con la API del backend.

3. Crear pantalla principal.

4. Consumir endpoint GET /test-catalog (Mongo: colección test_catalog) y mostrar catálogo.

5. Consumir endpoint GET /lab-order-events (Mongo: colección lab_order_events) y mostrar eventos (por lab_order_id o últimos).

6. Mostrar catálogo y eventos en listas (puede ser navegación simple).

7. Manejar estado de carga.

8. Manejar errores de conexión.

9. Evidenciar ejecución en emulador o dispositivo.

SISTEMAS OPERATIVOS – Ubuntu 24.04 (8 preguntas)

Objetivo: Ejecutar el sistema en Linux.

Trabajar desde el HOME (~).

1. Crear estructura del proyecto
   Crear una carpeta llamada examen, dentro de ella una carpeta laboratorio, y dentro de laboratorio crear las carpetas: backend, frontend, movil, docs. Luego verificar la estructura.
   Comandos a usar: cd, mkdir -p, tree

2. Navegación y verificación
   Entrar a la carpeta laboratorio, verificar la ubicación actual y listar contenido.
   Comandos a usar: cd, pwd, ls, ls -la

3. Crear archivos de evidencia
   Dentro de docs crear los archivos lab_comandos.txt y evidencia.txt, luego registrar la fecha actual dentro de evidencia.txt y verificar contenido.
   Comandos a usar: touch, date >>, cat

4. Redirección de salida
   Guardar la salida del comando who en un archivo y luego agregar la salida de ls -la al mismo archivo. Verificar contenido.
   Comandos a usar: who, ls -la, >, >>, cat

5. Buscar palabra dentro de archivo
   Dentro de la carpeta docs, crear o sobrescribir el archivo lab_comandos.txt con el siguiente contenido (copiar y pegar exactamente):

cat << EOF > lab_comandos.txt

Laboratorio - Backend

GET /api/lab-tests/
GET /api/lab-orders/
POST /api/lab-orders/
DELETE /api/lab-orders/10/
INFO: lab order created successfully
INFO: lab service running
WARN: lab orders timeout detected
EOF
Verificar que el contenido se guardó correctamente.
Luego buscar la palabra lab-orders dentro del archivo y mostrar también el número de línea donde aparece.
Comandos a usar: cat << EOF, cat, grep, grep -n

6. Localizar archivo
   Crear dentro de la carpeta frontend un archivo llamado README.md y luego localizarlo desde la carpeta laboratorio.
   Comandos a usar: touch, find, locate (opcional)

7. Copiar y mover archivos
   Copiar el archivo evidencia.txt para crear un respaldo y luego mover ese respaldo a otra carpeta del proyecto.
   Comandos a usar: cp, mv, ls -la

8. Permisos y Sticky Bit
   Crear una carpeta llamada shared_lab, aplicar permisos 1777 (Sticky Bit) y verificar que aparezca la letra t en los permisos.
   Comandos a usar: mkdir, chmod, ls -l