Guía de la Prueba de desempeño – Módulo 6.1

Java + Spring Boot

Nombre de la prueba: CoopCredit – Sistema Integral de Solicitudes de Crédito
Módulo de formación al que corresponde: Módulo 6.1 – Spring Boot
Duración de la prueba: La prueba tendrá un calendario semanal con la siguiente estructura para las jornadas AM y PM:

Hora	Día 1	Día 2	Día 3	Día 4	Día 5
06:00 – 07:00	Prueba técnica				
07:00 – 08:00					
08:00 – 08:20	Break 1	Sustentación de la solución entregada (individual)* *Con previa cita asignada*	Sustentación de la solución entregada (individual)* *Con previa cita asignada*	Feedback personalizado por parte de tu TL* *Con previa cita asignada*	Carga de resultados y feedback a Moodle por parte de tu TL*
08:20 – 09:00	Prueba técnica				
09:00 – 10:00					
10:00 – 10:20	Break 2				
10:20 – 11:00	Prueba técnica				
11:00 – 12:00					
12:00 – 12:20	Break 3				
12:20 – 13:00	Prueba técnica				
13:00 – 14:00					
Cambio de jornada					
14:00 – 15:00	Prueba técnica				
15:00 – 16:00					
16:00 – 16:20	Break 1	Sustentación de la solución entregada (individual)* *Con previa cita asignada*	Sustentación de la solución entregada (individual)* *Con previa cita asignada*	Feedback personalizado por parte de tu TL* *Con previa cita asignada*	Carga de resultados y feedback a Moodle por parte de tu TL*
16:20 – 17:00	Prueba técnica				
17:00 – 18:00					
18:00 – 18:20	Break 2				
18:20 – 19:00	Prueba técnica				
19:00 – 20:00					
20:00 – 20:20	Break 3				
20:20 – 21:00	Prueba técnica				
21:00 – 22:00					

Notas: **Durante **toda la jornada** tendrás trabajo **individual** asignado en el cual deberás estar avanzando, mientras eres llamado a tu cita de sustentación o feedback en la sala de entrenamiento habitual. Durante el **día 5** tu TL no estará disponible para resolver dudas, pero cuentas con tu tutor oficial.*

Temas abordados:

- Arquitectura Hexagonal (puertos, casos de uso, dominio puro, adaptadores)
- Spring Boot (REST, validaciones, AOP, ControllerAdvice)
- JPA + Hibernate avanzado (relaciones, transacciones, EntityGraph)
- Seguridad con Spring Security + JWT

- MapStruct para mapeo
- Observabilidad: Actuator + Micrometer + logging estructurado
- Pruebas (JUnit, Mockito, Spring Boot Test)
- Testcontainers
- Microservicios y comunicación REST
- Docker y docker-compose
- Flyway para migraciones

Herramientas y recursos:

Hardware requerido:
Laptop o computadora con conexión a internet (otorgada por Riwi)

Software requerido:

- IntelliJ IDEA o VS Code con extensiones de Java
- JDK 17 o superior
- Maven
- Spring Boot 3+
- PostgreSQL o MySQL
- Docker + Docker Compose
- Postman o Insomnia
- Git y GitHub
- Lombok
- MapStruct
- JUnit 5, Mockito
- Testcontainers

Recursos adicionales:
Documentación oficial de Spring Boot, Spring Security, JPA, Micrometer y Docker.

Apuntes, ejercicios y guías del módulo 6.

Objetivo de la prueba:

Identificar tus capacidades para aplicar los conocimientos teóricos y prácticos adquiridos en **Java SE**, desarrollando una aplicación de escritorio funcional para la gestión de bibliotecas, aplicando POO, persistencia con JDBC, manejo de excepciones, pruebas unitarias y documentación técnica.

La prueba busca identificar tu nivel de dominio en:

- Arquitectura Hexagonal real.
- Microservicios comunicándose entre sí.
- Seguridad robusta con JWT.

- Persistencia avanzada.
- Observabilidad y métricas.
- Pruebas unitarias y de integración.
- Dockerización completa.
- Documentar correctamente la aplicación con un README.md profesional.
- Defender técnicamente tu solución en una sustentación individual.

  El objetivo es construir:

1. credit-application-service

Microservicio principal encargado de:

- Gestión de afiliados.
- Gestión y evaluación de solicitudes de crédito.
- Integración con un servicio externo de riesgo.
- Control de roles y seguridad JWT.
- Manejo estandarizado de errores.
- Observabilidad y métricas.

2. risk-central-mock-service

Microservicio externo simulado, que devuelve un score basado en el documento.

Estructura de la prueba:

Esta prueba estará conformada por un **caso de uso real** en el que deberás resolver los problemas planteados por **CoopCredit**, simulando un entorno de desarrollo profesional y distribuido.

Caso de uso:
CoopCredit, una cooperativa de ahorro y crédito con sedes en varias ciudades, requiere un sistema integral que permita digitalizar y automatizar el proceso de solicitud y evaluación de créditos.

Parte 1 – Análisis y diseño de la solución

- Identificación de entidades:
  Affiliate, CreditApplication, RiskEvaluation, User (seguridad).
- Identificación de roles:
  ROLE_AFILIADO, ROLE_ANALISTA, ROLE_ADMIN.
- Análisis de flujos:
  - Registro de afiliado.
  - Registro de solicitud.
  - Evaluación automática usando risk-central.

- Diseño:
  - Diagramas:
    - Arquitectura hexagonal.
    - Diagrama de casos de uso.
    - Diagrama de microservicios.
  - Identificación de puertos:
    - RiskEvaluationPort (salida)
    - AffiliateRepositoryPort
    - CreditApplicationRepositoryPort
    - AuthPorts
  - Definición de casos de uso:
    - RegisterAffiliate
    - RegisterCreditApplication
    - EvaluateCreditApplication

Parte 2 – Implementación del Dominio y Persistencia JPA

- Modelado del dominio (POJOs sin anotaciones)
- Entidades JPA:
  - @OneToMany Affiliate → CreditApplications
  - @OneToOne CreditApplication → RiskEvaluation
- Reglas:
  - Validaciones: documento único, salario > 0, afiliado activo.
- Implementación de adaptadores de persistencia JPA.
- Migraciones con Flyway:
  - V1__schema.sql
  - V2__relations.sql
  - V3__initial_data.sql (opcional)
- Uso de:
  - EntityGraph
  - join fetch
  - batch-size
  - @Transactional en casos de uso críticos.

Parte 3 – Seguridad, Validaciones y Manejo Global de Errores

- Implementación de autentificación con **JWT stateless**.
- Registro + Login:
  - /auth/register
  - /auth/login
- Seguridad por roles:
  - Afiliado → solo sus solicitudes.

    - Analista → solicitudes pendientes.
    - Admin → acceso total.
- Validaciones cruzadas:
    - afiliado ACTIVO para solicitar crédito
    - cálculo de relación cuota/ingreso
    - monto máximo según salario
    - antigüedad mínima
- Manejo estandarizado de errores con:
    - ProblemDetail (RFC 7807)
    - @ControllerAdvice
- Logging estructurado.

Parte 4 – Microservicios, Integración y Observabilidad

- Construcción del microservicio `risk-central-mock-service`.
- Endpoint: POST /risk-evaluation
- Respuesta consistente por documento usando: hash como seed
- Integración desde credit-application-service usando un adapter REST.
- Observabilidad:
    - Actuator: health, metrics, info
    - Micrometer: métricas personalizadas
    - /actuator/prometheus (opcional)
- Métricas obligatorias:
    - tiempos por endpoint
    - errores
    - fallos de autenticación

Parte 5 – Pruebas, Docker y Entrega Final

- Pruebas unitarias:
    - Casos de uso puro del dominio
    - Mock del RiskEvaluationPort
- Pruebas de integración:
    - Spring Boot Test
    - MockMvc
    - Pruebas de seguridad
    - CRUD de solicitudes
- Testcontainers:
    - Base de datos PostgreSQL en contenedor
    - Pruebas reproducibles
- Docker:
    - Dockerfile multi-stage
    - docker-compose con:

- credit-application-service
- risk-central-mock-service
- db

- Documentación:
  - README profesional con:
    - arquitectura
    - diagrama hexagonal
    - instrucciones de ejecución
    - roles y flujos
    - colecciones de Postman

Permisos y limitaciones:

Se permite:

- Uso de documentación oficial (Spring, JPA, Security, Docker)
- Consulta de tus apuntes, ejercicios o materiales del módulo.
- Uso de IA bajo los lineamientos establecidos y aprobados por el TL.
- Uso de herramientas profesionales (IntelliJ, Docker, Postman)

No se permite:

- Comunicación con otros estudiantes durante la prueba.
- Copiar código o soluciones de terceros.
- Subir el código a repositorios externos durante la evaluación.

Integridad académica:
Cualquier forma de copia, plagio o uso de material no autorizado resultará en **calificación cero** y sanciones según las políticas internas de Riwi.

Idioma:
Todo el código, nombres de variables, clases, métodos y el README deben estar escritos en **inglés**. Solo los datos mostrados en interfaz o mensajes al usuario pueden estar en español.

Implicaciones:

- Esta prueba determina tu continuidad en el programa de entrenamiento según el Capítulo IV - Evaluación y calificación del Reglamento Interno de Riwi.
- Se evaluará tanto la solución técnica como tu defensa en la sustentación individual.

Evaluación:

Rúbrica: disponible en Moodle.
Se evaluarán los siguientes criterios:

- Arquitectura Hexagonal correcta
- Microservicios funcionando
- Seguridad JWT, Roles y Control de Accesos
- Validaciones + errores estándar
- Persistencia JPA
- Pruebas unitarias y de integración
- Testcontainers funcional
- Docker Compose operativo
- Documentación profesional
- Defensa técnica

Entrega:
Mediante la plataforma Moodle en el espacio asignado.
Deberás subir el enlace al repositorio GitHub y un archivo comprimido (.zip) del proyecto.

Visualización de resultados:
En Moodle al final del quinto día.

Lugar de retroalimentación:
Sala habitual de entrenamiento durante el cuarto día.

¡Éxitos en este evento evaluativo!