

## ÍNDICE GOLEARNIX

1. Introducción
  - a. Entorno y contextualización
  - b. Objetivos
2. Marco Conceptual (*aquí van todas las definiciones*)
3. Temporalización (*hablar de los sprints*)
  - a. Diario de planificación (*metodología ágil - scrum/kanban*)
  - b. Análisis
  - c. Investigación
  - d. Diseño
  - e. Codificación
4. Análisis, diseño, infraestructura y descripción
  - a. Descripción
  - b. Análisis
    - i. Requisitos iniciales
    - ii. Actores del sistema
    - iii. Diagrama de casos de uso (*con su descripción*)
  - c. Diseño
    - i. Diagrama de clases (*modelos e interfaces solo*)
    - ii. Diagrama de secuencia
    - iii. Bases de datos (*dividido por las dos aplicaciones y por las relacionales y no relacionales*)
      1. Modelado de datos
      2. Diagrama entidad-relación
    - iv. API first vs Code first
  - d. Infraestructura
    - i. Esquema de red
    - ii. Flujo de llamadas entre servidores durante las peticiones
5. Implementación del proyecto
  - a. Aplicación de autenticación
    - i. Arquitectura (*3 capas*)
      1. Arquitectura de sacrificio
      2. Principio KISS
    - ii. Primeros pasos con Go
    - iii. ¿Por qué usar UUID como identificador del usuario?
    - iv. Json Web Token
      1. Generación de tokens
      2. Sesiones
    - v. Conexión con la base de datos
    - vi. Gestor de eventos RabbitMQ
      1. Conexión
      2. Emisión de eventos
    - vii. Acciones definidas
      1. Crear usuario (registro)
      2. Inicio de sesión (login)
      3. Actualizar usuario
      4. Cerrar sesión

- 5. Eliminar usuario
  - viii. Documentación de la API con OpenAPI / Swagger
  - ix. Extracción de la configuración de `application.properties` a las variables de entorno
- b. Aplicación gestor de cursos
  - i. Arquitectura
    - 1. ¿Por qué usar la arquitectura hexagonal?
    - 2. Principios básicos de diseño
    - 3. Funcionamiento
  - ii. Proyectos multi-módulos con Maven
  - iii. Modelos de datos en dominio
  - iv. Modelos de datos en persistencia
    - 1. PostgreSQL
    - 2. Redis
  - v. Proyecciones
  - vi. Patrón CQRS
  - vii. Patrón repository
  - viii. Patrón saga
  - ix. Patrón assembler
  - x. Principio de modularización de configuración
  - xi. ¿Creación de canales en RabbitMQ?
  - xii. Acciones definidas
    - 1. Eliminar usuario
    - 2. Crear un curso
      - a. Manejo de relaciones
    - 3. Actualizar un curso
      - a. Manejo de relaciones
    - 4. Eliminar un curso
    - 5. Completar el progreso de una lección
    - 6. Inscribir a un usuario en un curso
  - xiii. Funcionamiento en microservicios de ambas aplicaciones
    - 1. Eventos
    - 2. Sistema de mensajería
  - xiv. Módulo de inicialización de datos (Patrón de diseño Data Seeding)
  - xv. Control de versiones de la base de datos con FlyWay
  - xvi. Documentación de la API con OpenAPI / Swagger
  - xvii. Extracción de la configuración de `application.properties` a las variables de entorno
- 6. Manual de usuario
  - a. Postman
  - b. Pruebas
- 7. Propuestas de mejoras
  - a. Clusters de alta disponibilidad
  - b. Separación en microservicios de la aplicación de gestión de cursos
  - c. Migración de eventos a Kafka con Debezium
  - d. Pruebas unitarias
  - e. Pruebas de integración
  - f. Despliegue con docker

8. Valoración personal
9. Puntos a destacar
10. Conclusión
11. Bibliografía (*siguiendo APA*)