Métodos/Técnicas de Ingeniería de Software
-- Evaluación de Reemplazo --

1. Descripción del trabajo

Los alumnos, en forma **personal**, deben desarrollar y desplegar una aplicación web diseñada en base a una arquitectura de microservicios.

2. Evaluación

- La nota final se calcula de la siguiente manera:

  PE = PROMEDIO(*CalidadProyectoProducto*, CalidadRespuestas)

- El detalle de cada uno de los parámetros de evaluación es el siguiente:
  - ***CalidadProyectoProducto****:* El/La alumno(a) deberá mostrar resultados en vivo (desde su computador) tanto de la gestión del proyecto como del desarrollo y despliegue del producto de software. El profesor es quien solicita que es lo que debe mostrar.
  - ***CalidadRespuestas:*** Nota que refiere a la calidad de las respuestas que el/la alumno(a) entregue cuando el profesor haga las preguntas dirigidas.

3. Lineamientos generales

- La evaluación se realizará en forma "**personal"**.
- Para la evaluación no se debe entregar ningún informe escrito.
- Cada alumno debe presentarse en forma puntual en la fecha/hora programada. En caso contrario se le calificará con la nota mínima 1.0
- A la evaluación solamente deben presentarse aquellos alumnos que fueron planificados para la fecha. No se permitirá el ingreso de otros alumnos.

4. Acerca del proyecto de software

4.1 Descripción del sistema

Desarrollar un sistema que permita llevar el control de préstamos de los data proyectores EPSON y ViewSonic que tiene el DIINF. Actualmente el DIINF tiene 7 data proyectores EPSON y 5 Data Proyectores ViewSonic. Básicamente se requiere gestionar los préstamos y devoluciones de estos 12 data proyectores. Para el caso de los préstamos se requiere registrar los siguientes datos: *fecha del préstamo*, *hora del préstamo*, *profesor al que se le hace el préstamo*, y *Uso que se le dará (Dictado de clases, Reuniones varias, Examen de título)*. Para el caso de las devoluciones se requiere registrar: *fecha de devolución*, *hora de devolución*, *estado de devolución (Buenas Condiciones, Con Daños)*.

Algunas reglas de negocio importantes son:

- Si un profesor devuelve un data proyector en estado "Con Daños" por tres veces queda inhabilitado para pedir data proyectores.
- Un data proyector no puede quedar en calidad de préstamo por más de 6 horas. Si un profesor devuelve un data proyector después de la 6 hrs., entonces no podrá pedir data proyectores hasta la próxima semana.
- Los data proyectores EPSON solamente pueden ser prestados para *Dictado de Clases* y *Exámenes de Título*.
- Los data proyectores ViewSonic solamente pueden ser prestados para *Reuniones Varias*.

El encargado de gestionar los préstamos de los data proyectores necesita que el sistema le muestre un reporte por pantalla donde pueda ver los préstamos hechos a los data proyectores. El sistema debe permitir seleccionar un data proyector y luego mostrar por pantalla "todos" los préstamos del data proyector seleccionado. El listado debe estar ordenado por fecha de préstamo y luego por hora de préstamo. Este reporte debe contener la siguiente información:

- Fecha préstamo
- Hora préstamo
- Profesor al que se le hizo el préstamo
- Fecha Devolución
- Hora de Devolución
- Número de Hrs en poder del profesor
- Estado en que fue devuelto
- Uso que se le dio al proyector

5. Aspectos del desarrollo del producto

5.1 Respecto del Frontend

- Debe ser desarrollado usando ReactJS.
- El frontend debe tener un buen diseño que asegure una adecuada usabilidad. Debe ser diseñado tomando en consideración las *10 Heurísticas* de Nielsen. *Nota: No se requiere hacer una evaluación de usabilidad con el cuestionario SUS.*
- Se requiere un único frontend para la aplicación.

5.2 Respecto del Backend

- Debe ser desarrollado usando el patrón arquitectural de microservicios. El sistema debe tener al menos 3 microservicios para implementar las siguientes funcionalidades:
    - Ingreso de data proyectores

    - Ingreso de profesores
    - Gestión de préstamos de data proyectores.
    - Gestión de devoluciones de data proyectores.
    - Reporte de préstamos/devoluciones
    - *NOTA: cada microservicio debe almacenar sus datos en su propia Base de Datos (MySQL o PostgreSQL).*
- Cada microservicio del *backend* debe ser desarrollado usando *Spring Boot* y usando una arquitectura de capas (@RestController, @Service, @Respository).
- El código fuente del *backend* debe ser escrito usando programación orientada a objetos.
- Todos los microservicios del *backend* deben usar **puertos estáticos**. No se aceptan soluciones con puertos dinámicos.
- El *backend* (además de tener los microservicios descritos anteriormente) debe tener implementado **"solamente"** los patrones de microservicios *ConfigServer* y *API Gateway*. **No debe** incluir *Service Discovery* (*Eureka Server*).

5.3 Despliegue del producto

- El *frontend* y todos los microservicios del *backend* deben ser empaquetados en contenedores independientes y luego almacenados en *Docker Hub*.
- El despliegue de la aplicación (tanto del *backend* como del *frontend*) se debe realizar usando *Docker Compose.*
- El despliegue de la aplicación web se debe realizar en el computador local o en un servidor de la nube (AWS, Azure, GCP, Digital Ocean, etc.)
- La aplicación web debe poder ser accedida desde un navegador web.