Métodos/Técnicas de Ingeniería de Software
-- Evaluación 3 --

1. Descripción del trabajo

Los alumnos, en forma **personal**, deben evaluar los aspectos funcionales y no funcionales de una aplicación web.

2. 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.

3. Acerca del proyecto de desarrollo

Se debe seleccionar el proyecto de la *Evaluación 1* o de la *Evaluación 2*, solo uno de ellos. Con el proyecto seleccionado se debe realizar todo lo indicado en la sección 4 de este documento. *IMPORTANTE: El proyecto seleccionado debe tener todas las funcionalidades (Épica 1 hasta la Épica 7) implementadas y funcionando correctamente.*

4. Aspectos del desarrollo de las pruebas

- ***Pruebas funcionales.*** Se requiere definir al menos 7 pruebas funcionales para las funcionalidades Épica 2 y Épica 6. Las pruebas funcionales deben ser redactadas en algún editor de textos (por ejemplo, MS Word) usando lenguaje Gherkin. De las pruebas funcionales definidas, al menos el 50% deben ser automatizadas en *Selenium IDE*. *IMPORTANTE: Cada prueba automatizada debe tener instrucciones que permitan saber si la prueba fue satisfactoria o no. Que estén solamente automatizadas no permite saber si la prueba fue satisfactoria o no.*
- ***Pruebas no funcionales:***
    - **Usabilidad:**
        - *Heuristicas de Nielsen*. Todo el *frontend* debe ser ajustado de tal manera que cumpla en forma satisfactoria con las ***diez*** *Heuristicas de Nielsen*.
        - *Cuestionario SUS.* Se requiere evaluar la usabilidad de la funcionalidad "Épica 2 – Gestión de Préstamos y Devoluciones" usando el cuestionario SUS. Se requiere un puntaje final SUS mayor o igual a 75. (Nota: se debe usar al menos 5 personas para evaluar). Usar *Microsoft Clarity (https://clarity.microsoft.com/)* para tener evidencias y así poder analizar cómo fue el comportamiento de cada usuario.
    - ***Rendimiento:*** Se deben realizar pruebas de rendimiento para las funcionalidades Épica 2 y Épica 6 usando *JMeter* y *Google Lighthouse*.
        - Para el caso de *Google Lighthouse,* se deben analizar las métricas *Performance*, *Accessibility*, y *Best Practices* para ambas épicas.

- Para el caso de *JMeter*, se deben usar los reportes *View Results in Table*, *Aggregate Report*, y *View Results Tree* para justificar los resultados. Se deben realizar las siguientes pruebas:
    - *Load testing.* Probar el sistema con diferentes cargas (usuarios concurrentes) para saber hasta dónde soporta sin caerse. Por ejemplo, 10 usuarios, 50 usuarios, 100 usuarios, 500 usuarios, 1000 usuarios, etc.
    - *Stress testing.* Probar el Sistema más allá de su capacidad normal, hasta que falle. Encontrar el punto de quiebre. Analizar qué sucede en ese punto.
    - *Volume Testing.* Probar el sistema con grandes volúmenes de datos en la BD. No es necesario muchos usuarios. ¿Cómo se comporta el sistema cuando las tablas de la BD llegan a tener grandes cantidades de datos?

- ***Mantenibilidad:*** Se debe realizar análisis estático del código fuente tanto del Frontend como del Backend (usando *Sonarqube)*. El análisis debe verificar lo siguiente:
    - El Backend debe tener un ratio de deuda técnica (Technical Debt Ratio) menor o igual a 1.0%.
    - El Frontend debe tener un ratio de deuda técnica (Technical Debt Ratio) menor o igual a 2.0%.
    - El Backend no debe presentar los siguientes code smells: *Long Method, Large Class, Long Parameter List, Inappropriate Comments, Dead Code, Duplicate Code*.
    - El Backend debe cumplir con el "*Google Java Style Guide*" [https://google.github.io/styleguide/jsguide.html].
    - El Frontend debe cumplir con "*Airbnb JavaScript Style Guide*" [https://javascript.airbnb.tech/].