Proyecto DevOps

El objetivo del obligatorio es la aplicación práctica de los conceptos de DevOps. Se implementarán prácticas de CI/CD, QA y testing automático sobre una base de código existente. El proyecto que se usará como base utiliza lenguajes y tecnologías del semestre anterior de la carrera.

Objetivos.

- Aplicar un marco de gestión ágil.
- Analizar la deuda técnica.
- Implementar un repositorio y procedimientos de versionado.
- Crear un pipeline con eventos y acciones.
- Integrar prácticas de QA en el pipeline y gestionar el feedback.
- Generar escenarios de testing desde la perspectiva del usuario.
- Automatizar el testing funcional o de caja negra.
- Reflexionar sobre DevOps.

Entregas y esfuerzo.

La ejecución del proyecto tiene 5 entregas de duración variable entre una y dos semanas, con la confección final de un informe académico que resuma los resultados y aprendizajes del proyecto. En cada entrega se aplicarán ceremonias de la gestión ágil que el equipo deberá incorporar y adaptar.

El proyecto se realizará en el marco de gestión Kanban. Se espera que cada integrante dedique un mínimo de 5 horas semanales a las actividades de ingeniería del proyecto (sin incluir gestión y retrospectivas). Antes de cada entrega, el equipo realizará una retrospectiva y confeccionará un informe de avance, que debe incluir:

- Registro de las actividades realizadas (fecha, horas, integrante).
- Resultados obtenidos en el período.
- Dificultades encontradas y formas de solución.
- Lecciones aprendidas y mejoras en el proceso.

Se debe realizar un informe académico que detalle las actividades realizadas y justifique las decisiones tomadas para cumplir con estos objetivos. **Es esencial el fundamentar todas las decisiones tomadas durante el proyecto.**

La retrospectiva deberá considerar algún método conocido (por ej. DAKI) y su análisis debe estar basado en datos.

Producto a construir/modificar.

En su oportunidad el equipo docente hará entrega del producto de software (ya construido) que cada uno de los equipos deberán mantener, ya sea por reparación de bugs o extensión de funcionalidades.

Rúbrica con detalle de ejecución.

Proyecto DevOps			
Entregas (en semanas)	**Actividades principales**	**Artefactos recomendados**	**Puntaje Max.**
Primera (2 sem)	• Definición de marco general de KANBAN, primera versión del proceso de ingeniería, creación del repositorio y definición de roles del equipo (PO, SM, DESA, TES, etc.). Todos los integrantes del equipo deben ser si o si DESA y TES, independientemente que tengan otro rol. • Análisis de deuda técnica. • Realización de una retrospectiva, basada en el método DAKI.	• Pequeña guía de "Definición/uso del proceso de ingeniería en el contexto de KANBAN". • Pequeña guía de "Creación y posterior mantenimiento del repositorio: elementos que contiene y cómo los van a versionar". • Análisis de deuda técnica: (1) issues ingresados en GitHub con su descripción y su respectiva clasificación (bugs - severidad -, mejoras, problemas de estándares, etc.) y (2) un documento resumen de las issues. • Informe de avance de la etapa. • Detalle de registro de esfuerzo por tipo de tarea, según las actividades del proceso de ingeniería o del tablero, si se prefiere. • Totales de registro de esfuerzo por la entrega. • Video de la retrospectiva con el “SM”.	**7**
Segunda (1 sem)	• Creación de primera versión del tablero (en GitHub) en función del proceso de ingeniería enmarcado en KANBAN. • Configuración del pipeline en función del tablero antes mencionado. • Elección de dos bugs registrados como issues en GitHub, fundamentando la elección. Los bugs deben ser de la mayor severidad.	• Pequeña guía de "Explicación del tablero y su vínculo con el proceso de ingeniería". • Pequeña guía de "Configuración del pipeline y su vínculo con el tablero". • Pipeline en GitHub Actions configurado. • Informe con identificación y justificación de los bugs en función la clasificación proporcionada (véase Anexo1).	**5**

	• Reparación de ambos bugs utilizando TDD. • Realización de una retrospectiva, basada en el método DAKI. • Realización de una revisión de los bugs con el PO.	• Código de software reparado. • Código de casos de prueba. • Evidencia de ejecución de casos de prueba. • Informe de avance de la etapa. • Detalle de registro de esfuerzo por tipo de tarea, según las actividades del proceso de ingeniería o del tablero, si se prefiere. • Totales de registro de esfuerzo por la entrega. • Video de la retrospectiva con el “SM”. • Video de la revisión de los bugs con el PO.	
Tercera (2 sem)	• Actualización del proceso de ingeniería (segunda versión) basado en BDD, enmarcado en KANBAN. • Creación de la segunda versión del tablero (en GitHub) en función del proceso de ingeniería enmarcado en KANBAN, mencionado anteriormente. • Configuración del pipeline en función del tablero antes mencionado. • Creación de escenarios de prueba en BDD para las nuevas funcionalidades. • Desarrollo del **front-end y solo del front-end** de las nuevas funcionalidades. • Desarrollo del **back-end y solo del back-end** de las nuevas funcionalidades. • El desarrollo y la prueba de ambas funciones debe ser desarrollado en el orden indicado anteriormente en forma estricta. • Realización de una retrospectiva, basada en el método DAKI.	• Nueva versión de pequeña guía de "Definición/uso del proceso de ingeniería en el contexto de KANBAN". • Nueva versión de pequeña guía de "Explicación del tablero y su vínculo con el proceso de ingeniería". • Nueva versión de pequeña guía de "Configuración del pipeline y su vínculo con el tablero". • Nueva versión del pipeline en GitHub Actions, si es necesario. • Código de software extendido. • Código de casos de prueba. • Evidencia de ejecución de casos de prueba. • Informe de avance de la etapa. • Detalle de registro de esfuerzo por tipo de tarea, según las actividades del proceso de ingeniería o del tablero, si se prefiere. • Totales de registro de esfuerzo por la entrega.	**8**

	• Realización de una revisión de las nuevas funcionalidades con el PO.	• Video de la retrospectiva con el “SM”. • Video de la revisión de las nuevas funcionalidades con el PO.	
Cuarta (1 sem)	• Automatización del test exploratorio (funcional) de ambos bugs y ambas nuevas funcionalidades. • La herramienta para esta prueba debe ser Selenium. • Confección y análisis de métricas DevOps. • Realización de una retrospectiva, basada en el método DAKI.	• Evidencia de ejecución de casos de prueba. • Informe de avance de la etapa. • Detalle de registro de esfuerzo por tipo de tarea, según las actividades del proceso de ingeniería o del tablero, si se prefiere. • Totales de registro de esfuerzo por la entrega. • Métricas de DevOps. • Video de la retrospectiva con el “SM”. • Video de la revisión de los bugs con el PO.	**4**
Quinta (2 sem)	• Realización de informe final académico.	• Informe académico final.	**6**
		Total	**30**

Notas:

1. El informe de avance de cada etapa, incluyendo la quinta, se entregará subiendo un archivo PDF a Aulas. Se debe entregar al terminar cada etapa del proyecto hasta las 23:59 del martes indicado. Si el día de entrega coincide con un feriado no laborable, la misma puede prorrogarse hasta un día no feriado.
2. El informe académico final debe entregarse, además, por el sistema de gestión de ORT el día 20 de junio del 23. Los estudiantes deben estar inscriptos y formar equipo de obligatorio en el sistema. Si no se cumple con los requisitos administrativos de la universidad no se puede corregir el obligatorio.
3. Una vez finalizada la quinta entrega se seleccionarán grupos para realizar una defensa del trabajo.

Plataforma tecnológica básica.

- Tableros, pipelines, repositorios y CI/CD: GitHub.
- BDD: Specflow.
- Test exploratorio o funcional: Selenium.
- Desarrollo: .NET / Angular.

Plataforma tecnológica extendida.

Se espera que el equipo investigue y aplique técnicas y herramientas sobre el pipeline que permitan mejorar el valor entregado al cliente, en el contexto tecnológico específico del proyecto, tales como:

- Análisis estático de código y linters.
- Ejecución automática de tests unitarios.
- Análisis de cobertura de código.
- Revisión de código mediante pull request.
- En forma opcional, se pueden aplicar en el proyecto estrategias de despliegue en la nube y otras herramientas de QA (testing de performance, testing de seguridad, etc.).

Sobre las nuevas funcionalidades.

Las user stories a desarrollar serán comunicadas oportunamente por Aulas.

Anexo1: Clasificación de bugs.

1. Según su prioridad.
   Definición de prioridad: urgencia o importancia en reparar el defecto, asociada a la planificación, lo cual implica precedencia. La fija el Product Owner.

   Clasificación:
   - Inmediata: plazo máximo 24 hs. (generalmente los defectos de severidad crítica se asocian a prioridad inmediata).
   - Alta: plazo máximo 48 hs.
   - Media: plazo máximo un par de semanas.
   - Baja: sin plazo.

2. Según su severidad.
   Definición de severidad o gravedad: es el impacto que genera el bug sobre el sistema. Refiere también a la adherencia a estándares e implica afectación del sistema. La fija el Tester.
   Clasificación:
   - Critico: un defecto que obstaculice o bloquee completamente la prueba o el uso de un producto o función.
   - Mayor: una función principal que no cumpla con los requisitos y se comporte de manera diferente a lo esperado. Cuando funciona muy lejos de las expectativas o no está haciendo lo que debería estar haciendo.
   - Menor: función que no cumpla con sus requisitos y se comporte de manera diferente a lo esperado, pero su impacto es insignificante hasta cierto punto o no tiene un impacto importante en la aplicación.
   - Leve: defecto cosmético.

Anexo2: Guía para la confección del informe académico final.

Se debe realizar un informe académico que detalle las actividades llevadas a cabo y justifique las decisiones tomadas para cumplir con los objetivos del Proyecto DevOps.

Se espera que el informe académico sea una síntesis global del proyecto DevOps.

El informe académico debe considerar tanto los informes de avance como las reflexiones sobre los aprendizajes durante el proyecto, de acuerdo con los objetivos planteados en la letra del obligatorio, los que se muestran en la Rúbrica.

Este informe no debe incluir detalles operativos ni documentos de trabajo, los cuales, conjuntamente con otros artefactos del proyecto, deben estar contenidos en el repositorio y referenciados desde este.

El informe debe estar enfocado en las lecciones aprendidas, a través de reflexiones sobre las prácticas de ingeniería de software aplicadas y su impacto, así como mostrar las conclusiones del trabajo realizado.

Se sugiere que la longitud de este informe académico no supere las 10 páginas, excepcionalmente se podría admitir un informe de hasta quince páginas.

Recuérdese que es esencial el fundamentar todas las decisiones tomadas durante el proyecto.

Este informe debe estar conformado por distintos apartados que consideren los ítems que se detallan más abajo. Es importante destacar que estos ítems no pretenden, y por ende, no tienen que ser apartados del informe, sino una guía para lo que los estudiantes confeccionen el informe. En resumen: los ítems a continuación no son necesariamente apartados del informe.

1. Resumen de gestión del proyecto.

Se debe incluir un resumen de la gestión de proyecto que agrupe los cuatro informes de avance. Se deberán explicitar, entre otros, los pasos o actividades realizadas, productos entregados o entregables de cada actividad, cantidad de issues reportadas y cerradas y **métricas del proyecto (este aspecto es fundamental).**

2. Reflexiones sobre el aprendizaje.

Considerando cada uno de los objetivos de la Rúbrica, el equipo deberá reflexionar sobre sus aprendizajes y resultados obtenidos durante el proyecto. Las reflexiones sobre cada objetivo no deberían superar una página.

Como elemento orientativo para el planteamiento de estos aprendizajes y resultados se sugiere utilizar algo similar al método DAKI: Drop (¿Qué hicimos mal y deberíamos dejar de hacer?), Add (¿Qué no hicimos y deberíamos comenzar a hacer?), Keep (¿Qué hicimos bien y deberíamos continuar haciendo?) y, finalmente, Improve (¿Qué hicimos relativamente bien, pero podríamos mejorar?).

3. Lecciones aprendidas.

Las lecciones aprendidas podrían ser resultantes del paso anterior y podrían encararse como sugerencias o “tips” a, por ejemplo, nuevas generaciones que fueran a enfrentarse a situaciones similares de un proyecto parecido a este de DevOps.

Para orientar mejor el planteamiento de una lección aprendida, podrían escribir el enunciado de la lección y luego la descripción de algún ejemplo o situación o anécdota que el equipo efectivamente experimentó, respecto de la lección.

4. Conclusiones.

Las conclusiones del informe deberían poder responder a las siguientes interrogantes:

- ¿Qué significan los resultados de lo investigado, aplicado y/o solucionado?
- ¿Qué consecuencias/implicancias tiene lo anterior?
- ¿Por qué son importantes esas implicancias o consecuencias?
- ¿A dónde nos conducen?

5. Guía de instalación para desarrollo y despliegue en producción.

El informe debe incluir una guía resumida que ayude a la instalación del código y dependencias para realizar el build, ejecutar los casos de prueba unitarios, casos de prueba funcionales, etc. La guía debe considerar los pasos para desplegar el sistema en producción (sin necesidad de abrir el IDE).

Téngase en cuenta que los docentes se basarán en esta guía para ejecutar el pipeline y probarlo.