

<b>EVALUACIÓN</b>	Obligatorio	<b>GRUPO</b>	71390 N6A	<b>FECHA</b>	19/09/2024
<b>MATERIA</b>	Certificado en DevOps				
<b>CARRERA</b>	Analista en Tecnologías de la Información   Actualización profesional				
<b>CONDICIONES</b>	<p> <b>- Puntaje máximo:</b> 55 puntos  <b>- Puntaje mínimo:</b> 1 punto  <b>- Fecha de entrega:</b> 09/12/2024 hasta las 21:00 horas en <a href="http://gestion.ort.edu.uy">gestion.ort.edu.uy</a> (max. 40Mb en formato zip, rar o pdf) </p> <p><b>Uso de material de apoyo y/o consulta</b></p> <p><u>Inteligencia Artificial Generativa</u></p> <ul style="list-style-type: none"> <li>- Seguir las pautas de los docentes: Se deben seguir las instrucciones específicas de los docentes sobre cómo utilizar la IA en cada curso.</li> <li>- Citar correctamente las fuentes y usos de IA: Siempre que se utilice una herramienta de IA para generar contenido, se debe citar adecuadamente la fuente y la forma en que se utilizó.</li> <li>- Verificar el contenido generado por la IA: No todo el contenido generado por la IA es correcto o preciso. Es esencial que los estudiantes verifiquen la información antes de usarla.</li> <li>- Ser responsables con el uso de la IA: Conocer los riesgos y desafíos, como la creación de “alucinaciones”, los peligros para la privacidad, las cuestiones de propiedad intelectual, los sesgos inherentes y la producción de contenido falso</li> <li>- En caso de existir dudas sobre la autoría, plagio o uso no atribuido de IAG, el docente tendrá la opción de convocar al equipo de obligatorio a una defensa específica e individual sobre el tema</li> </ul> <p><b>IMPORTANTE:</b></p> <ol style="list-style-type: none"> <li>1) Inscribirse</li> <li>2) Formar grupos de hasta 2 personas del mismo dictado</li> <li>3) Subir el trabajo a Gestión antes de la hora indicada (ver hoja al final del documento: “RECORDATORIO”)</li> </ol> <p>Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con el Coordinador o Coordinación adjunta <b>antes de las 20:00hs.</b> del día de la entrega, a través de los mails <a href="mailto:alamon@ort.edu.uy">alamon@ort.edu.uy</a> y <a href="mailto:fernandez_ma@ort.edu.uy">fernandez_ma@ort.edu.uy</a>, o telefónicamente al 29021505 - int 1156 u 1138</p>				

## 1. Presentación del problema

En el corazón de la transformación digital de la empresa líder en retail, emergió un desafío inesperado, uno que pondría a prueba la resiliencia y adaptabilidad de su recién adoptada metodología de trabajo. La rápida transición hacia prácticas innovadoras y la adopción de nuevas tecnologías revelaron una brecha significativa en la comprensión y la comunicación entre los equipos de desarrollo y operaciones.

Este desafío se manifestó durante el lanzamiento de una nueva aplicación destinada a revolucionar la experiencia de compra de los clientes. A pesar de los esfuerzos meticulosos y la implementación de herramientas de vanguardia, el equipo se encontró frente a un problema recurrente: los despliegues de nuevas versiones de la aplicación resultaban en errores inesperados y caídas del sistema, afectando directamente la experiencia del usuario y la reputación de la empresa.

La raíz del problema no era técnica, sino cultural y organizativa. La tradicional separación entre los equipos de desarrollo, encargados de innovar y agregar nuevas características a la aplicación, y el equipo de operaciones, responsable de mantener la estabilidad y disponibilidad del sistema, había creado un ambiente donde la responsabilidad compartida y la comunicación efectiva eran deficientes.

El equipo de desarrollo, enfocado en la rapidez y en la entrega continua de nuevas funciones, a menudo subestimaba el impacto de sus cambios en la infraestructura y en la experiencia del usuario final. Por otro lado, el equipo de operaciones, priorizando la estabilidad sobre la innovación, se encontraba constantemente en una posición reactiva, lidiando con problemas sin tener una comprensión completa de las nuevas actualizaciones.

Este escenario evidenció un problema fundamental: la falta de una cultura de colaboración y entendimiento mutuo entre los equipos. La solución requería más que solo la adopción de herramientas y prácticas; necesitaba un cambio en la mentalidad y en la forma en que los equipos interactuaban entre sí.

Ante la creciente tensión y los desafíos que surgían de la desconexión entre los equipos de desarrollo y operaciones, la dirección ejecutiva de la compañía intervino, reconociendo la necesidad de abordar no solo los síntomas sino las causas fundamentales de estos problemas. Se hizo evidente que la solución requerirá más que ajustes superficiales en los procesos técnicos; siendo imperativo un cambio cultural profundo que alinearé a todos los equipos bajo un conjunto común de objetivos y prácticas.

Por lo tanto, se le solicita al equipo de proyecto un plan de acción detallado que no solo aborde las ineficiencias operativas evidentes sino que también fomente un ambiente de colaboración, transparencia y aprendizaje continuo.

Este plan deberá incluir estrategias específicas para mejorar la comunicación y la colaboración entre los equipos de desarrollo y operaciones, identificar y eliminar barreras que impidan la eficacia de los flujos de trabajo integrados, establecer prácticas que promuevan una comprensión mutua de los desafíos y objetivos compartidos.

Se espera que, a través de esta iniciativa, la empresa no sólo superará los obstáculos actuales sino que también sentará las bases para una agilidad y resiliencia operativa a largo plazo, asegurando así su posición competitiva en el mercado.

## 2. Entrega

- a. Utilizar un tablero Kanban para realizar la planificación y seguimiento de las tareas identificadas a realizar por el equipo.
- b. El equipo de trabajo deberá analizar y desarrollar los ciclos de integración y delivery continuo (**CI/CD**) de al menos tres ambientes, por ejemplo: Dev, Test y Prod. Los ciclos no podrán contener menos de tres (3) etapas/tareas.
- c. El equipo deberá de realizar los siguientes deployments:
  - i. Empaquetar las cuatro (4) aplicaciones de **backend (BE)** en containers y desplegarlas en un **orquestador o motor de containers** en la nube de AWS (o nube de preferencia).
  - ii. Seleccionar y desplegar una (1) aplicación **frontend (FE)** de las tres (3) provistas por el equipo docente sobre el servicio S3 bucket de AWS (o nube de preferencia).
- d. Aplicar algún test sobre las aplicaciones de BE (4 en total) o la aplicación de FE elegida (pruebas de carga, pruebas automatizadas con Postman, Selenium, etc) y registrar los resultados. (prueba extra en roadmap)
- e. Aplicar alguna herramienta de análisis de código estático (para todas las aplicaciones) realizar un informe sobre los resultados obtenidos y recomendaciones a implementar para mejorar la calidad del mismo.
- f. Almacenar todo el código generado (para el trabajo de DevOps como el utilizado para los microservicios y aplicativo de FE) en repositorios de Git, determinando una estrategia de ramas acorde a las etapas del ciclo de desarrollo si es que lo ameriten (**Git Flow o Trunk Based**). Además, se deberá trabajar con el flujo de trabajo de [feature branch](#) (el flujo de trabajo solamente para la parte de DevOps), evidenciando el trabajo de los miembros del equipo. Se recomienda que generen una organización en la herramienta de versionado que elijan para alojar los repositorios de Git (por si luego quieren dar acceso de lectura al equipo docente). En caso de elegir GitHub como plataforma, los usuarios del equipo docente son los siguientes:
  - i. [ElLargo](#)
  - ii. [mauricioamendola](#)
  - iii. [cheredia2k12](#)
- g. Deberán de subir la entrega final al repositorio brindado por el equipo docente (el cual se encuentra en la organización con la identificación de ambos estudiantes), en gestión suben lo mismo pero en formato .zip o .rar, siguiendo el siguiente orden de estructura:
  - i. Readme: Este Readme file deberá de contener la información detallada de la implementación realizada. **Cuanto mejor documentada se encuentre la misma y explicativa sea la documentación, mejor.**
  - ii. Carpetas varias: Deben de alojar en el repositorio los archivos que fueron obteniendo a medida que hicieron el trabajo (IaC, CI/CD, diagramas, evidencia de tablero Kanban, etc.). Queda a elección de cada equipo como manejar esta estructura de carpetas. **No es de interés que alojen el código de las aplicaciones de BE y FE.**

- 
- h. Se deberán de realizar los diagramas para los flujos de integración de código para la parte de desarrollo y la parte de DevOps.
  - i. Documentar toda la implementación o información de interés usando Markdown o AsciiDoc dentro del mismo repositorio de GitHub (brindado por el equipo docente, nombrado en el punto **g.**), **dentro de la rama main/master**.
  - j. Realizar los diagramas correspondientes a los procesos de CICD.
  - k. Toda la infraestructura disponibilizada en la nube de AWS (o nube de preferencia), deberá de ser manejada como **Infrastructure as Code (IaC)**.
  - l. Realizar alguna tarea con servicios serverless (AWS Lambda o API GW). La tarea a realizar queda a decisión del equipo dependiendo el tipo de servicio serverless elegido. Ejemplo de tareas a realizar:
    - i. AWS Lambda: Crear alguna automatización que puede ir desde respuestas a ciertos eventos, CI/CD, monitoreo y alertas, gestión de IaC, automatización de backups, análisis de logs y respuestas automatizadas, seguridad y cumplimiento, respuesta a cambios de estado en servicios.
    - ii. API GW: Conectar los servicios de BE para que queden solamente accesibles mediante el API GW con el fin de brindarles una capa de seguridad extra.

---

## Algunas consideraciones / aclaraciones

1. Las aplicaciones a desplegar (tanto de backend a empaquetar como aplicación web) serán provistas por los docentes. Las de backend serán objeto de estudio en el taller de Microservicios. Las mismas se encuentran dentro del repositorio del curso:
  - a. [FE](#)
  - b. [BE](#)
2. Las aplicaciones de FE y BE no tienen comunicación entre sí, las de BE si, como fue evidenciado en el taller de Microservicios.
3. Queda a consideración del equipo, la elección de las herramientas a desplegar / utilizar, el equipo docente dejará un listado disponible con posible recomendaciones de herramientas a utilizar.
4. Se deberá exponer el trabajo realizado en una instancia de defensa. Para esto, recomendamos usar una presentación de venta a modo de guía. La presentación de venta consta en la venta de la solución/producto, **por ende, no es una explicación técnica.**
5. Queda a elección del grupo, el formato de la documentación, sabiendo que será valorada la **prolijidad, calidad y organización** del mismo. (Readme.md del punto g.i)
6. El equipo deberá entregar el contenido almacenado en el repositorio de Git (de DevOps) en el formato establecido por el sistema de gestión. (.rar o .zip).
7. Se podrán realizar TODAS las consultas que quieran, mientras estas sean CLARAS, CONCISAS y tengan un mínimo de investigación previa. **No se podrán enviar documentos para ser corregidos o medir el avance fuera de las fechas de entrega previstas.**

## 3. Defensa

Como defensa deberán realizar una presentación del proyecto realizado mediante Teams / Zoom en una sesión grupal de no más de 20 minutos. Es necesario tener la cámara encendida y el micrófono abierto. Podremos llamar de forma individual, solo si es necesario, a un alumno para complementar lo expuesto en la defensa grupal.

#### 4. Rúbrica

SE PIDE	PUNTAJE MÁXIMO	SATISFACTORIO
La solución propuesta por el equipo tiene que cumplir con contemplar todos los puntos propuestos por el equipo docente y tener un grado de innovación propuesto por el equipo.	7	La solución cumple con todos los puntos necesarios para ser innovadora, atractiva y cubrir todas las problemáticas del cliente.
Deben de manejar el tablero de Kanban para poder hacer seguimiento del trabajo a tener que realizar para lograr el objetivo.	4	Se visualiza la buena utilización por parte del equipo sobre tablero de Kanban durante el transcurso del proyecto.
Deben presentar un documento en Markdown o AsciiDoc que evidencie toda la documentación respectiva al trabajo realizado. En el repositorio que utilicen.	4	<ul style="list-style-type: none"> <li>- No contenga faltas de ortografía.</li> <li>- El estilo de redacción es correcto para una documentación técnica.</li> <li>- El formato es consistente (se mantienen fuentes, estilos, tamaños de letras, etc).</li> </ul>
Desplegar la aplicación backend de manera empaquetada sobre algún servicio manejador de contenedores.	4	Utilizar algún servicio de AWS para alojar los contenedores (o de la nube de preferencia).
Desplegar la aplicación de frontend sobre algún servicio serverless.	4	Utilizar algún servicio de AWS alojar la aplicación de FE (o de la nube de preferencia).

Utilizar repositorios de Git para alojar todo el código manejado para las diferentes fases del proyecto.	4	Se tienen repositorios separados y bien identificados para cada una de las partes del código del proyecto (desarrollo, código de DevOps, etc).
Manejo del flujo de trabajo de feature branch para el/los repositorio/s correspondiente al código de DevOps, con el fin de fomentar la revisión cruzada y aumentar la calidad del código obtenido.	4	Se evidencia el uso de feature branch con pull requests en el repositorio (de DevOps) por parte de ambos miembros del equipo con comentarios en los casos que se ameriten.
Tener uno de los servicios serverless desplegados con un uso deseable.	4	Se evidencia uno de los servicios serverless configurado.
Toda la infraestructura disponibilizada en el proveedor de nube deberá de ser implementada como IaC.	4	Se evidencia el uso de alguna herramienta de IaC.
Uso de alguna herramienta de análisis de código estático y realizar un informe sobre los resultados obtenidos.	4	Se presenta un informe detallado con los problemas detectados y acciones correctivas a implementar.
Uso de alguna herramienta para realizar test sobre la aplicación y presentar los resultados obtenidos.	4	Se presenta un informe detallado de las pruebas ejecutadas y sus resultados.
Definición de o los GitFlows o trunk bases a manejar para los repositorios.	4	Se presentan los diagramas correspondientes a los diferentes GitFlows o trunk based.
Se deben implementar ciclos de CI/CD con al menos tres etapas.	4	Se presentan los ciclos de CI/CD para el código de las aplicaciones (y DevOps en caso que lo ameriten).



## 5. Roadmap sugerido

### **Semana 1:**

1. Familiarizarse con la letra y empezar a identificar tareas a realizar. Bajando las mismas en el tablero de Kanban.
  - a. Las tareas del tablero van a ir mutando a medida que vayan pasando las semanas, por lo cual van a crear/modificar/eliminar tareas en todo momento.
2. Definición de diagramas de flujos de integración para las diferentes partes y realizar los diagramas.
3. Investigación y elección para las siguientes partes:
  - a. Herramienta de Git.
  - b. Herramienta de CICD.
  - c. Orquestador.
  - d. Cloud provider.
  - e. Herramienta para análisis de código estático.
  - f. Herramienta para análisis de prueba extra.
  - g. Elección de aplicativo de FE a buildear y desplegar.
  - h. Elección del servicio serverless a usar y cual va a ser la finalidad de uso.
4. Armar estructura de repositorios en la herramienta de Git con los diferentes componentes de código. Impactar las definiciones de los diagramas de flujos de integración correspondiente a cada repositorio.

### **Semana 2:**

1. Empaquetar aplicaciones de BE de manera local, validar que funcione todo de manera correcta localmente, para luego empezar a realizar el mismo proceso en la herramienta de CICD (parte de CI).
2. Construcción del aplicativo de FE de manera local, validar que funcione todo de manera correcta localmente, para luego empezar a realizar el mismo proceso en la herramienta de CICD (parte de CI).
3. Empezar a familiarizarse con Terraform (o herramienta de IaC elegida) para comenzar a escribir la IaC necesaria.

### **Semana 3:**

1. Terminar de implementar infraestructura con Terraform (o herramienta de IaC elegida).
2. Empezar a realizar la parte de CD para las aplicaciones de BE y FE.
3. Agregar etapa de análisis de código estático para pipelines de BE y FE.
4. Empezar a realizar la configuración de servicio serverless.
5. Agregar etapa de prueba extra elegida en la semana uno en el/los pipeline/s correspondiente/s.
6. Empezar a armar la presentación correspondiente a la defensa. Elegir que tipo de demo se va a realizar.

#### **Semana 4:**

1. Realizar diagramas correspondientes a los diferentes flujos de CICD.
2. Realizar informes correspondientes a las pruebas (punto d. y e. de la entrega)
3. Terminar de pulir detalles correspondientes a puntos anteriores.
4. Terminar de armar la documentación de entrega final.
5. Terminar de armar la presentación correspondiente a la defensa.
6. Subir la entrega final al repositorio brindado por el equipo docente y en gestión.

## **6. Documentación de ejemplo**

Se dejan las siguientes documentaciones de semestres anteriores a modo de **EJEMPLO**. Usenlas como guías, realicen su propia impronta de documentación, cualquier evidencia de copia/plagio, será sancionable por el equipo docente.

- [Ejemplo1](#)
- [Ejemplo2](#)
- [Ejemplo3](#)
- [Ejemplo4](#)
- [Ejemplo5](#)

## **7. Presentación de venta de ejemplo**

Se deja la siguiente [presentación](#) de venta de los alumnos Xian Harding Inglés y Santiago Langwagen del curso 2022S2.

## **8. Herramientas sugeridas**

- Planificación:
  - Jira
  - Trello
  - Asana
  - Azure Boards
  - Monday.com
  - ClickUp
- Code:
  - GitHub
  - GitLab
  - Bitbucket
  - Azure Repos
  - SourceForge

- Phabricator
- AWS CodeCommit
  
- Build:
  - Jenkins
  - GitLab CI/CD
  - GitHub actions
  - Travis CI
  - CircleCI
  - TeamCity
  - Apache Maven
  - Gradle
  - Azure DevOps
  - AWS CodePipeline/CodeBuild
  - Docker
  - Bitbucket
  
- Test:
  - Selenium
  - JUnit
  - Postman
  - TestNG
  - SonarQube
  - SonarCloud
  - Blackduck
  - Fortify
  - Cypress
  - Mocha
  - Chai
  
- CI/CD:
  - Jenkins
  - GitLab CI/CD
  - GitHub actions
  - CircleCI
  - Azure DevOps
  - AWS CodePipeline
  - Travis CI
  - Spinnaker
  - Bamboo
  - Argo CD
  - Bitbucket

- Deploy:
  - Kubernetes
  - Docker
  - Ansible
  - Terraform
  - AWS CodeDeploy
  - OpenShift
  - Rancher
  - Nomad
  - Chef
  - Puppet
  
- Cloud:
  - AWS (Amazon Web Services)
  - Azure
  - Google Cloud Platform (GCP)
  - DigitalOcean
  - IBM Cloud
  - Oracle Cloud
  - Alibaba Cloud


---

## RECORDATORIO: IMPORTANTE PARA LA ENTREGA

- **Obligatorios**

La entrega de los obligatorios será en formato digital online, a excepción de algunas materias que se entregarán en Bedelía y en ese caso recibirá información específica en el dictado de la misma.

Los principales aspectos a destacar sobre la **entrega online de obligatorios** son:

1. Ingresá al sistema de Gestión.
2. En el menú, seleccioná el ítem “Evaluaciones” y la instancia de evaluación correspondiente, que figura bajo el título “Inscripto”.
3. Para iniciar la entrega hacé clic en el ícono: 
4. Ingresá el número de estudiante de cada uno de los integrantes y hacé clic en “Agregar”. El sistema confirmará que los integrantes estén inscriptos al obligatorio y, de ser así, mostrará el nombre y la fotografía de cada uno de ellos. Una vez agregados todos los integrantes, hacé clic en “Crear equipo”.

**Cualquier integrante podrá:**

- **Modificar la integración del equipo.**
- **Subir el archivo de la entrega.**

5. Seleccioná el archivo que deseás entregar. Verificá el nombre del archivo que aparecerá en la pantalla y hacé clic en “Subir” para iniciar la entrega. Cada equipo (hasta 2 estudiantes) debe entregar **un único archivo en formato zip o rar** (los documentos de texto deben ser pdf, y deben ir dentro del zip o rar). El archivo a subir debe tener **un tamaño máximo de 40mb**

Quando el archivo quede subido, se mostrará el nombre generado por el sistema (1), el tamaño y la fecha en que fue subido.

6. El sistema enviará un e-mail a todos los integrantes del equipo informando los detalles del archivo entregado y confirmando que la entrega fue realizada correctamente.
7. Podés cerrar la pestaña de entrega y continuar utilizando Gestión o salir del sistema.
8. La **hora tope para subir el archivo será las 21:00** del día fijado para la entrega.
9. La entrega se podrá realizar desde cualquier lugar (ej. hogar del estudiante, laboratorios de la Universidad, etc).
10. Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con la Coordinadora o Coordinación adjunta antes de las 20:00hs. del día de la entrega, a través de los mails, [alamon@ort.edu.uy](mailto:alamon@ort.edu.uy) o [fernandez\\_ma@ort.edu.uy](mailto:fernandez_ma@ort.edu.uy); o telefónicamente al 29021505 - int 1156 u 1138