Introducción

Siguiendo las distintas filosofías Cloud Native un equipo de tecnología puede ofrecer soluciones de negocio diseñadas para ambientes altamente cambiantes que requieren de un alto desempeño. Para construir este tipo de aplicaciones no solo basta conocer los conceptos y los principios que las forman, sino también entender de manera práctica su funcionamiento y los retos derivados de su diseño e implementación.

Este proyecto tiene como propósito acercar al estudiante a la implementación de este tipo de aplicaciones, haciendo uso de tecnologías de vanguardia y simulando situaciones reales de trabajo en equipo. Se espera que el estudiante pueda identificar e implementar las mejores prácticas para diseñar y desarrollar aplicaciones nativas en la nube cumpliendo con altos estándares de calidad.

Para lograrlo, los estudiantes diseñarán e implementarán una aplicación siguiendo unas primeras decisiones de arquitectura, y aplicarán patrones, técnicas y principios de las aplicaciones nativas para cumplir con los requerimientos que serán exigidos en el contexto del problema.

Objetivos

Poner en práctica los conceptos presentados en el curso.
Diseñar y desarrollar aplicaciones nativas en la nube que sigan buenas prácticas de ingeniería de software y de arquitectura.
Identificar consideraciones técnicas y métricas de negocio que deben ser tenidas en cuenta para prestar un servicio altamente escalable capaz de soportar millones de usuarios.
Implementar una aplicación altamente escalable y disponible soportada en los servicios de los proveedores de nube, entendiendo las implicaciones de arquitectura, complejidad técnica e impacto financiero.
Comprender los requerimientos y aspectos técnicos que deben ser tenidos en cuenta para crear aplicaciones escalables, resilientes y extensibles.

Contexto general del proyecto

Le recomendamos realizar la lectura de los siguientes documentos en el orden que se indica antes de revisar el enunciado de la entrega:

Metodología

El proyecto tiene un total de 3 entregas. El proceso de desarrollo es incremental por lo que cada semana se avanza en el desarrollo y se incorporarán las prácticas y las tecnologías que se van estudiando en el curso. Cada entrega contará con desarrollo, despliegue, documentación y mecanismos para pruebas de cada uno de los componentes desarrollados.

Las entregas del proyecto serán desarrolladas en equipos de 4 personas y el trabajo deberá ser gestionado en GitHub a través de las herramientas de gestión de proyectos provistas.

Aunque el proyecto tendrá 3 entregas distribuidas en el tiempo de duración del curso, se espera que cada semana el equipo cumpla con ciertas actividades, las cuales se describirán en los enunciados de cada entrega.

Las entregas están distribuidas de la siguiente manera:

Entrega 1: semana 3

Durante las primeras semanas del curso, el equipo construirá una versión del sistema haciendo uso de herramientas de contenerización y orquestación sobre un entorno local, la cual debe cumplir con algunos requerimientos de negocio y de arquitectura definidos. Esta entrega solo tendrá en cuenta los temas vistos durante las semanas 1 y 2.

La fecha límite de envío para esta entrega será el sábado de la tercera semana a las 11:59 pm GMT - 5.

Entrega 2: semana 6

Para esta entrega el equipo aplicará patrones de diseño y nuevas tecnologías, agregando nuevos componentes y funcionalidades al sistema. Esta entrega utilizará los componentes de la entrega 1 y aplicará los temas vistos durante las semanas 3 y 4.

La fecha límite de envío para esta entrega será el sábado de la sexta semana del curso a las 11:59 PM GMT - 5.

Entrega 3: semana 8

Para esta entrega el equipo se concentrará en agregar nuevos componentes a la aplicación haciendo uso de tecnologías sin servidor y patrones de arquitectura orientada a eventos. Para esta entrega se tendrá en cuenta los temas vistos hasta la semana 7.

La fecha límite de envío para esta entrega será el sábado de la octava semana del curso a las 11:59 PM GMT - 5.

> Los criterios de calificación, así como las reglas para la presentación y posterior calificación, se encuentran en la sección de reglas del curso.

Introducción

En esta sección se describe de forma general el proyecto que usted trabajará con su equipo, así como los conceptos básicos a tener en cuenta para cada una de las entregas.

Modelo de negocio

> Se recomienda hacer una lectura general de este documento, identificando los actores del problema, los principales requisitos deseados del sistema. Posteriormente, es recomendable hacer una lectura más detallada, para identificar elementos en detalles específicos del problema planteado.

En los últimos años se han presentado diversos cambios en las dinámicas sociales debido a múltiples factores globales y locales, siendo la pandemia el precursor de un crecimiento totalmente inesperado, y que ha acelerado los cambios que estaban estimados para los próximos 5 años, en tan solo uno.

Sectores cómo el comercio electrónico y logística se han vuelto en las principales herramientas para que millones de personas adquieran bienes y servicios durante el último año; tan solo Colombia presentó un incremento de usuarios de hasta 113% en el 2023 en plataformas como Mercado Libre, y hubo más de 66 mil millones de dólares en transacciones en toda Latinoamérica en el mismo año.

Se prevé que el crecimiento y adopción del comercio electrónico continúe, principalmente, debido a que el grado de confianza en el modelo se ha incrementado, lo que abre la posibilidad a que nuevas empresas, usuarios, productos y servicios se sumen a este ecosistema en los próximos años.

Teniendo esto presente, un grupo de emprendedores quiere aprovechar esta oportunidad para tratar de ganar un espacio en el sector de logística y envío de productos, ofreciendo un servicio que consideran diferenciado y que puede consolidarse como una fuerte alternativa frente a sus competidores. Planean crear un servicio de intermediario (broker) para el envío de paquetes con un modelo de economía colaborativa.

Este servicio tiene en cuenta un solo usuario que puede realizar dos tipos de actividades: Arrendador y Arrendatario. Como Arrendador el usuario desea aprovechar el espacio disponible en las maletas de viajero de otro usuario para hacer envío de paquetes, y como Arrendatario o pasajero, desea ofrecer un espacio en su maleta a cambio de una ganancia económica. Ambos usuarios se benefician económicamente del servicio y la compañía gana una comisión por servir de intermediario. Un usuario puede crear publicaciones en la plataforma indicando el trayecto en el que puede llevar un envío. Los otros usuarios pueden revisar las publicaciones y ofertar por un espacio en la maleta describiendo que desean enviar y el espacio necesario.

> En una primera versión de la aplicación, toda la comunicación y organización entre las partes (Arrendador y Arrendatario) se realizará de manera externa a la aplicación.

Este grupo de emprendedores desea convertir la plataforma en un fuerte competidor de delivery de paquetes, tanto nacional como internacional, estimando un total de 1 millón de envíos, 15 millones de usuarios registrados y transacciones por más de 10 millones de dólares al finalizar su quinto año de operación.

Para lograrlo, se ha conseguido una inversión semilla para contratar a un equipo de desarrollo de alto nivel, quienes construirán una versión inicial de la aplicación en los próximos dos meses. Esta primera versión permitirá demostrar la viabilidad del negocio y avanzar en las próximas rondas de inversión.

La misión de su equipo es construir una aplicación siguiendo los requerimientos y diseño inicial del sistema, y desplegarla haciendo uso de servicios de la nube, una arquitectura de microservicios y una serie de tecnologías ya seleccionadas por el equipo fundador. Al finalizar, el equipo habrá logrado construir una aplicación que siga un diseño que beneficie atributos de calidad como escalabilidad, extensibilidad, portabilidad, resiliencia y disponibilidad.

Glosario

Para el desarrollo de este proyecto, tenga presente las siguientes definiciones.

> Trayecto: especifica el punto de inicio y el punto final de un viaje. Un trayecto tiene una fecha y un costo de envío de una maleta.
> Publicación: es un aviso digital que indica que un usuario tiene disponibilidad para llevar un encargo en un trayecto en unas fechas en específico.
> Oferta: es la solicitud que hace un usuario en la publicación de otro, para hacer envío de un encargo en un trayecto. La oferta contiene una descripción del producto, un tamaño (`voluminoso`, `pequeño`, `mediano`), si es delicado o no y un monto en dólares que se propone por su envío.
> Utilidad ó Score: es una ganancia aproximada sobre una oferta realizada. Esta se calcula de la forma: `utilidad = monto oferta - (porcentaje de ocupación de una maleta * valor de la maleta en el trayecto)`. El porcentaje de ocupación de la maleta se define por la siguiente tabla:

Tamaño paquete	**Porcentaje ocupación en la maleta**
VOLUMINOSO / LARGE	100%
MEDIANO / MEDIUM	50%

PEQUEÑO / SMALL	25%

Entrega 1

Comprende los temas vistos en la semana 1 y 2 y su fecha límite es el sábado de la semana 3 11:59 pm GMT - 5.

Table of contents

Enunciado de la entrega 1

Para poder realizar esta entrega, es importante que cuente con lo siguiente:

> Acceso a la organización en GitHub con su correo Uniandes.
> Pertenecer a un equipo y contar con un repositorio de trabajo para el proyecto, el cual será creado por los tutores.

Si aún no dispone de estos elementos, le recomendamos solicitarlos lo antes posible. Tenga en cuenta que los retrasos en el acceso a los servicios del curso no podrán ser considerados como justificación para entregas fuera de plazo.

TABLE OF CONTENTS

Objetivos

> Evaluar la capacidad del equipo para implementar y desplegar una aplicación multicontenedor, aplicando buenas prácticas de desarrollo y calidad de código.
> Desplegar la solución localmente usando un orquestador de contenedores.
> Asegurar la calidad del código y los componentes desarrollados mediante integración continua.

Al finalizar esta entrega, su equipo habrá implementado una aplicación multicontenedor que se ejecuta en un orquestador de contenedores y ofrece un API REST para gestionar usuarios, publicaciones, ofertas y trayectos.

Prerrequisitos y restricciones

La entrega debe cumplir con las restricciones descritas en el documento de restricciones y utilizar las tecnologías listadas en el documento de tecnologías.

Alcance

Durante esta primera entrega, el equipo de ingeniería debe implementar y probar los componentes base del sistema sobre los cuales se construirán futuras funcionalidades. Se espera que el equipo:

- Desarrolle las aplicaciones de acuerdo con la arquitectura y API especificadas en la vista funcional.
- Implemente la lógica de negocio y modelos de datos según la vista de información.
- Prepare los archivos de despliegue en Kubernetes con base en la vista de despliegue.
- Incluya pruebas unitarias que validen el comportamiento de las aplicaciones.
- Documente técnicamente el sistema, siguiendo el enfoque de Documentación como Código.

> Las decisiones de arquitectura son parte fundamental del curso y deben seguirse tal como están definidas.

Los entregables de esta entrega son:

Documentación

Crear la documentación técnica tomando como base la documentación brindada en este proyecto y haciendo uso de las herramientas de Documentación como Código vistas en el curso.

- Archivo `README.md` en la raíz del repositorio indicando la estructura de las carpetas y un paso a paso para hacer el despliegue de todos sus componentes.

Archivo `README.md` en cada carpeta de aplicación. En este archivo se describe la estructura, pruebas, variables de ambiente, configuración y ejecución de la aplicación. (Ejemplo de un archivo README).
Documentación técnica del sistema escrita en Markdown y ubicada en la carpeta `/docs`. La estructura de la documentación debe ser la siguiente:
    - Página de inicio donde se presente al equipo y las tecnologías seleccionadas para el desarrollo (lenguajes, librerías de pruebas, manejo de dependencias, ejecución, desarrollo, etc), y la tabla de contenido para navegar a cada vista de arquitectura.
    - Vista de información: Describe los datos que maneja el sistema e incluye un modelo de entidades que está implementado por medio de PlantUML y validado correctamente en el pipeline. Tome como base el contenido de la vista de información, el modelo de entidades debe presentar los atributos (nombre, tipo) de cada entidad.
    - Vista funcional: Describe brevemente los componentes desarrollados e incluye un modelo de componentes implementado por medio de PlantUML y validado correctamente en el pipeline. Tome como base el modelo que se presenta en la vista funcional.
    - Vista de despliegue: Describe como los componentes de la vista funcional están siendo desplegados y los conectores entre ellos. Contiene un modelo de despliegue y un modelo de red tomando como base la actual vista de despliegue pero reimplementada por medio de PlantUML y validada correctamente en el pipeline.
    - Vista de desarrollo: Describe las decisiones de desarrollo para el desarrollo del proyecto, entre ellas:
        - Estructura de las carpetas y del proyecto.
        - Tabla de tecnologías donde se describen las herramientas de desarrollo, ejecución, pruebas y despliegue.
    - Las imágenes y archivos de PlantUML deben estar alojados en la carpeta `docs/diagrams/` y deben seguir la siguiente nomenclatura:
        - `components.puml` contiene el modelo de componentes de la vista funcional.
        - `deployment.puml` contiene el modelo de despliegue de la vista de despliegue.
        - `entities.puml` contiene el modelo de entidades de la vista de información.
        - `networks.puml` contiene el modelo de red de la vista de despliegue.
En la raíz del proyecto DEBE existir un archivo yaml con el nombre `config.yaml` con la siguiente estructura:

```
team:
  name: <nombre del equipo>
  members:
    - user: <usuario uniandes>
      percentage: <0 a 100>
    - user: <usuario uniandes>
      percentage: <0 a 100>
    - user: <usuario uniandes>
      percentage: <0 a 100>
    - user: <usuario uniandes>
      percentage: <0 a 100>
board: <link al dashboard de tareas del equipo>
docs: <link al github pages del equipo>
users_app:
  folder: <nombre de la carpeta donde está la aplicación de usuarios>
  image_name: <nombre de la imagen de la aplicación de usuarios, mismo que el nombre en el archivo de despliegue de k8s>
  image_tag: <tag de la imagen de la aplicación de usuarios, por ejemplo: latest, v1.0.0. Debe ser el mismo que el tag en el archivo de despliegue de k8s>
  authors:
    - <usuario uniandes>
posts_app:
  folder: <nombre de la carpeta donde está la aplicación de publicaciones>
  image_name: <nombre de la imagen de la aplicación de publicaciones, mismo que el nombre en el archivo de despliegue de k8s>
  image_tag: <tag de la imagen de la aplicación de publicaciones, por ejemplo: latest, v1.0.0. Debe ser el mismo que el tag en el archivo de despliegue de k8s>
```

```
  authors:

    - <usuario uniandes>

offers_app:

  folder: <nombre de la carpeta donde está la aplicación de ofertas>

  image_name: <nombre de la imagen de la aplicación de ofertas, mismo que el nombre en el archivo de despliegue de k8s>

  image_tag: <tag de la imagen de la aplicación de ofertas, por ejemplo: latest, v1.0.0. Debe ser el mismo que el tag en el archivo de despliegue de k8s>

  authors:

    - <usuario uniandes>

routes_app:

  folder: <nombre de la carpeta donde está la aplicación de trayectos>

  image_name: <nombre de la imagen de la aplicación de trayectos, mismo que el nombre en el archivo de despliegue de k8s>

  image_tag: <tag de la imagen de la aplicación de trayectos, por ejemplo: latest, v1.0.0. Debe ser el mismo que el tag en el archivo de despliegue de k8s>

  authors:

    - <usuario uniandes>
```

> El archivo `config.yaml` es el archivo más importante en el repositorio. Este archivo contiene la configuración que se usa en los pipelines para evaluar su entrega y se define la calificación de cada miembro del equipo. Tenga en cuenta que, si este archivo no está correctamente configurado, no será posible calificar la entrega.

Formato de entrega

- Archivos markdown en el repositorio del equipo en el release `ProyectoPrimeraEntrega`.
- Página web desplegada en github pages y creada a partir de los archivos en la carpeta `/docs`.

> Este entregable se califica sobre un release que debe ser creado en el repositorio de su equipo y que debe llevar el nombre `ProyectoPrimeraEntrega`. El release debe ser creado antes de la hora límite de entrega del proyecto. Si el release es creado posterior a la hora límite, se aplicarán las sanciones descritas en la sección de calificación del curso. La calificación se realizará con base en el contenido del release marcado como `ProyectoPrimeraEntrega`, por lo que cualquier cambio fuera de este no será considerado.

Mecanismo de evaluación

Este entregable se evaluará de dos formas:

- Por medio de pipelines del repositorio. El su repositorio encontrará el pipeline `ci_evaluador_docs.yml`, el cual debe ejecutarse de manera exitosa sobre el último commit de su release. Si este archivo es modificado el entregable será penitenciado como indica la rúbrica.
  Manualmente, los tutores del curso revisarán a detalle su página web para validar que todas las unidades y los modelos son correctos. Tenga presente que la página web de su proyecto debe ser privada.

Implementación

Implementar cuatro aplicaciones del sistema (usuarios: users, publicaciones: posts, ofertas: offers, rutas o trayectos: routes) los cuales están definidas en las vistas de arquitectura. Las aplicaciones deben ser ejecutadas correctamente en un ambiente local y en los pipelines por medio de Minikube. Para esta entrega el despliegue en un proveedor de nube NO está en el alcance.

- Código funcional de las cuatro aplicaciones base del sistema. Revise la vista funcional para tener un contexto general de como funcionan esta cuatro aplicaciones.
  - Todo el código debe estar almacenado en el repositorio que se le ha asignado a su equipo.
  - Cada application es un componente independiente y puede estar implementada en el lenguaje que desee: Python, Nodejs, Java, etc. Nuestra recomendación es trabajar con Python dada su experiencia en otros cursos con frameworks como Flask y FastAPI.
  - Cada aplicación debe cumplir con su definición de API. Revise la siguiente documentación donde se encuentran los detalles de como deben construirse los componentes y el API que debe tener cada uno.

- API de Usuarios
- API de Trayectos
- API de Publicaciones
- API de Ofertas

Cada aplicación tiene una propia base de datos y los datos que maneja cada una de las aplicaciones se encuentran descritos en la vista de información. El manejo de la información debe respetar la cohesión, por lo que una aplicación NO debe por ninguna razón almacenar o manipular la información que corresponde a otra. Revise la vista de despliegue para comprender como deben estar desplegados los componentes en esta entrega.

Todo el código debe ser entregado sobre la rama principal master/main. NO se revisará código en otras ramas.

Cada aplicación debe estar en su propia carpeta respetando la nomenclatura de nombramiento de componentes que se presentan en la vista funcional.

En cada carpeta se encuentra el código fuente, sus pruebas unitarias, un archivo `README.md` con la descripción del componente y el archivo `Dockerfile`. (Ejemplo de un archivo README).

Las aplicaciones deben debe ser desplegadas en contenedores haciendo uso de Docker y orquestadas por medio de Minikube. Revise la vista de despliegue para comprender como deben estar desplegados los componentes en esta entrega.

Cada componente será una aplicación independiente (código en carpeta diferente con sus propias dependencias y configuración) que se ejecutará cada una en un contenedor separado y tendrá una relación con una base de datos (una base de datos por cada aplicación). La base de datos debe ejecutarse en otro contenedor, NO debe hacerse uso de servicios externos en la nube.

En la raiz del repositorio se debe encontrar la carperta `/k8s` la cuál contiene los archivos necesarios para ejecutar todas las aplicaciones con sus bases de datos sobre Minikube.

Con el fin de incrementar la cohesión, disminuir el acoplamento e incrementar la seguridad, las bases de datos de las aplicaciones deben estar aisladas por medio de políticas de red. Las políticas deben estar diseñadas para que únicamente las aplicaciones dueñas de cada base de datos pueda conectarse a ellas. Por ejemplo, la base de datos `users_db` solo debe recibir conexiones de la aplicación `users_app`.

Los volúmenes que se deben usar para la entrega no son persistentes, sino efímeros. Su propósito radica en evitar que ante una indisponibilidad

momentanea del contenedor de base de datos se pierdan los datos. Haga uso de volúmenes tipo emptyDir.

Formato de entrega

Link del release `ProyectoPrimeraEntrega` creado antes de la fecha y hora máxima de entrega.

Este entregable se califica sobre un release que debe ser creado en el repositorio de su equipo y que debe llevar el nombre `ProyectoPrimeraEntrega`. El release debe ser creado antes de la hora límite de entrega del proyecto. Si el release es creado posterior a la hora límite, se aplicarán las sanciones descritas en la sección de calificación del curso. La calificación se realizará con base en el contenido del release marcado como `ProyectoPrimeraEntrega`, por lo que cualquier cambio fuera de este no será considerado.

Mecanismo de evaluación

Este entregable se evaluará de dos formas:

Por medio de pipelines. En su repositorio encontrará el pipeline `ci_evaluador_entrega1_k8s.yml`, el cuál debe ejecutarse de manera exitosa sobre el último commit de su release. Si este archivo es modificado el entregable será penitenciado como indica la rúbrica. El pipeline realiza las siguientes tareas:

- Verifica que la configuración de K8s es correcta validando que se crean los pods, volúmenes, servicios, redes, etc.
- Verifica que las políticas de red están correctamente configuradas.
- Ejecuta pruebas de API sobre cada aplicación para validar que la aplicación si responde al API definido. Para validar el aislamiento entre aplicaciones, el pipeline bloquea el tráfico a las demás aplicaciones mientras prueba cada una.

Manualmente se verificará que la solución entregada sea acorde a las decisiones de diseño planteadas en el enunciado: Componentes, despliegue, estructura. Se evaluará la cohesión y aislamiento; que una aplicación no gestione o tenga acceso a información que debe ser gestionada por otras aplicaciones.

Pruebas

Implementar pruebas unitarias que cubran mínimo el 70% de cada aplicación.

El código debe contar con pruebas unitarias que cubran al menos 70% del código de cada aplicación. La ejecución de las pruebas se validará únicamente en pipelines. Si las pruebas no son ejecutadas y la cobertura no es verificada por el pipeline el entregable será penitenciado como indica la rúbrica. NO se admiten ejecuciones locales.
Se debe modificar el archivo de pipelines con el nombre `ci_evaluador_unit.yml` para que contenga todos los jobs de pruebas unitarias. Cada job debe estar diseñado para probar el cubrimiento de una aplicación, y este no debe ser menor al 70% (En el código fuente del repositorio de su equipo encontrará un ejemplo de un proyecto en Python y como está configurado un pipeline de github).

Formato de entrega

Link del release `ProyectoPrimeraEntrega` creado antes de la fecha y hora máxima de entrega.

Este entregable se califica sobre un release que debe ser creado en el repositorio de su equipo y que debe llevar el nombre `ProyectoPrimeraEntrega`. El release debe ser creado antes de la hora límite de entrega del proyecto. Si el release es creado posterior a la hora límite, se aplicarán las sanciones descritas en la sección de calificación del curso. La calificación se realizará con base en el contenido del release marcado como `ProyectoPrimeraEntrega`, por lo que cualquier cambio fuera de este no será considerado.

Mecanismo de evaluación

Este entregable se evaluará únicamente por medio de pipelines del repositorio. El archivo `ci_evaluador_unit.yml` debe contener mínimo 4 jobs (uno por aplicación) en los que se verifica que el cubrimiento de pruebas es exitoso.

Si el equipo utiliza otro lenguaje diferente a Python, u otro framework de pruebas distinto a pytest, deben agregar una sección en su README describiendo las herramientas que usaron, como funcionan, donde se encuentra la documentación oficial de las tecnologías, y describir ese funcionamiento en su video de sustentación.

Presentación

Grabar un video donde presente su equipo, los detalles más relevantes de su proyecto, y una demo de como funciona el sistema.

> En el video se presenta al equipo que trabajó en la entrega, una descripción del repositorio y un demo de como funciona la aplicación haciendo uso de un cliente como Postman. La duración del video debe estar entre cinco y diez minutos. En caso de exceder el tiempo máximo, se aplicará una penalización en la nota de este entregable.

Formato de entrega

El video debe ser incluido en la descripción de su release `ProyectoPrimeraEntrega` en el momento de la creación. El video no puede ser incluido posterior a la creación del release y NO será recibido por ningún otro medio o mecanismo.

Mecanismo de evaluación

El video se evaluará manualmente verificando que describe la estructura del sistema y presenta una demo del funcionamiento de la aplicación. Si el video no cumple con la duración esperada o su calidad no permite evaluar que su contenido es el esperado, el entregable será penitenciado como indica la rúbrica.

Calificación

La calificación se hará tomando el éxito de inspecciones manuales y el éxito de los pipelines ejecutados. Tenga presente que el tutor puede, si lo considera necesario, ejecutar las pruebas en otros ambientes y tomar el resultado de esa ejecución como su nota. Será debidamente notificado en el caso que suceda.

La calificación se realizará tomando como base esta rúbrica.

> Después de la entrega, uno o dos grupos podrán ser seleccionados aleatoriamente por los tutores para realizar una sustentación síncrona en la que deberán exponer su solución y cada uno de los entregables descritos; la nota será calculada con base en esa presentación. Esa sustentación sucederá en los siguientes tres días hábiles a la fecha de entrega.

Estrategia recomendada

En esta sección se indica cuál es la estrategia recomendada que usted con su equipo deberían de seguir para cumplir con el entregable. Esta solo es una recomendación, es decisión de su equipo las tareas a desarrollar semana a semana.

División de trabajo

Recomendamos que cada integrante del equipo trabaje en un componente/aplicación lo que incluye código, configuración de la base de datos, contenedores,s pipelines, archivos de despliegue en k8s, pruebas y documentación.

¿Qué debería hacer la semana 1?

Revisar el enunciado y las restricciones.
Formar el equipo.
Planear de las actividades y división de responsabilidades. Haga uso del tablero Kanban de su repositorio para planear y gestionar su trabajo.
Definir las tecnologías y la estructura de las aplicaciones.
Actualizar el archivo `config.yaml`, al menos la parte de las aplicaciones.
Crear la documentación técnica con base en la documentación actual.
Construir las aplicaciones en contenedores, incluyendo las pruebas. Para manejar la conexión a base de datos podría correr la base de datos cómo otro contenedor y gestionar su ejecución directamente con Docker o Docker Compose, o ejecutar una base de datos directamente en su máquina física.

¿Qué debería hacer la semana 2?

Completar la construcción de las aplicaciones, incluyendo las pruebas.
Completar la configuración del pipeline de integración continua (pruebas unitarias).
Construir los archivos de k8s que permita el despliegue sobre su cluster de minikube Minikube.

¿Qué debería hacer la semana 3?

Completar el despliegue con Minikube y verificar que funciona tanto localmente, como en los pipelines.
Agregar volúmenes, redes y probar que funciona correctamente.
Completar la documentación, archivos de configuración y demás entregables.
Crear el video y el release.

Recursos:

Usted cuenta con talleres y ejemplos en el contenido del curso. Otros recursos que pueden ser de utilidad para la entrega 1 los puede encontrar en la página Recursos entrega 1

Recomendaciones finales

Revise la rúbrica antes de entregar.
Asegúrese de que todos los pipelines se ejecuten exitosamente.
Valide que la documentación sea clara y completa.
Recuerde que el trabajo en equipo y la planificación son clave para el éxito del proyecto.
Pregunte si tiene duda, no asuma cosas.

Rúbrica

Rúbrica de la entrega 1

Esta rúbrica depende totalmente del archivo `config.yaml` el cuál es el archivo más importante en el repositorio. Este archivo contiene la configuración que se usa en los pipelines para evaluar su entrega y se define la calificación de cada miembro del equipo. Si este archivo no está correctamente configurado, su entrega no puede ser calificada y su nota será de cero.

Entregable	Elemento	Máxima calificación	Criterio de calificación	Calificación por criterio
Presentación	Video.	5	El video se encuentra en la descripción del release, cumple con el tiempo definido y presenta una demo del funcionamiento satisfactorio de todas las aplicaciones haciendo uso de la colección de postman.	
			El video se encuentra en la descripción del release, presenta parcialmente el funcionamiento de las aplicaciones o supera el tiempo permitido.	3
			No se hace la entrega del video o no presenta el funcionamiento de las aplicaciones.	
Documentación	El archivo README.md del proyecto	2	El archivo README.md contiene toda la información de la estructura del proyecto, como se ejecuta y prueban todas las aplicaciones.	
			El archivo README.md está incompleto o no se encuentra en el repositorio.	

	Página web de documentación técnica.	5	La página web de la documentación del proyecto contiene todas las páginas esperadas incluyendo las 4 vistas con sus diagramas correspondientes generados usando PlantUML. La documentación es correcta y corresponde a las vistas de la primera entrega. Cada página corresponde a 20% de la nota, si alguna página no está presente o está incompleta, se penitenciará con su porcentaje completo.	
	Pipeline de documentación	2	Los pipelines de verificación de la documentación se ejecutan satisfactoriamente y no han sido modificados.	
			Los pipelines de verificación de la documentación fueron modificados o fallaron.	
Implementación	K8s - Componentes de despliegue	12	La configuración de k8s para el despliegue de las aplicaciones de forma declarativa es correcta y despliega correctamente las 4 aplicaciones, sus servicios, sus bases de datos y sus volúmenes. Se calcula con base en el porcentaje de éxito del pipeline ci_evaluador_entrega1_k8s.yml.	

			En el caso que el pipeline haya sido modificado la nota será de 0.	
	K8s - Redes	6	La configuración de políticas de red es correcta: las políticas se crean correctamente limitando la comunicación entre aplicaciones y bases de datos por el puerto definido en las restricciones. Se calcula con base en el porcentaje de éxito del pipeline ci_evaluador_entrega1_k8s.yml. En el caso que el pipeline haya sido modificado la nota será de 0.	
	Aplicación de gestión de usuarios.	13	La aplicación de gestión de usuarios cumple con las pruebas del API ejecutándose de manera independiente de las otras aplicaciones y por medio de Docker y Minikube. Se calcula con base en el porcentaje de éxito del pipeline de pruebas automáticas ci_evaluador_entrega1_k8s.yml que cubren los endpoints del servicio de usuarios.	

	Aplicación de gestión de trayectos.	13	La aplicación de gestión de trayectos cumple con las pruebas del API ejecutándose de manera independiente de las otras aplicaciones y por medio de Docker y Minikube. Se calcula con base en el porcentaje de éxito del pipeline de pruebas automáticas ci_evaluador_entrega1_k8s.yml que cubren los endpoints del servicio de trayectos.	
	Aplicación de gestión de publicaciones.	13	La aplicación de gestión de publicaciones cumple con las pruebas del API ejecutándose de manera independiente de las otras aplicaciones y por medio de Docker y Minikube. Se calcula con base en el porcentaje de éxito del pipeline de pruebas automáticas ci_evaluador_entrega1_k8s.yml que cubren los endpoints del servicio de publicaciones.	
	Aplicación de gestión de ofertas.	13	La aplicación de gestión de ofertas cumple con las pruebas del API ejecutándose de manera independiente de las otras aplicaciones y por medio de Docker y Minikube. Se calcula con base en el porcentaje de éxito del pipeline de pruebas automáticas ci_evaluador_entrega1_k8s.yml	

			que cubren los endpoints del servicio de ofertas.	
Pruebas	Pruebas unitarias de la aplicación de gestión de usuarios.	4	El código de gestión de usuarios tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el pipeline ci_evaluador_unit.yml.	
			El código de gestión de usuarios no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	
	Pruebas unitarias de la aplicación de gestión de publicaciones.	4	El código de gestión de publicaciones tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el pipeline ci_evaluador_unit.yml.	
			El código de gestión de publicaciones no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	

	Pruebas unitarias de la aplicación de gestión de ofertas.	4	El código de gestión de ofertas tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el pipeline ci_evaluador_unit.yml.	
			El código de gestión de ofertas no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	
	Pruebas unitarias de la aplicación de gestión de trayectos.	4	El código de gestión de trayectos tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el pipeline ci_evaluador_unit.yml.	
			El código de gestión de trayectos no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	

Restricciones

Las siguientes restricciones se tienen que cumplir en la implementación. El incumplimiento de estas restricciones implica la no calificación de las entregas.

> Este documento puede estar siendo actualizado durante el desarrollo del curso, cualquier cambio será comunicado previamente a todos los estudiantes por los canales definidos.

Restricciones generales

Este proyecto se trabajará única y exclusivamente en los repositorios brindados por la dirección del curso. Si no ha sido agregado al repositorio, favor comuníquese con el tutor por medio de slack.
La estrategia para el proyecto es Monorepo. Haga uso de un solo repositorio de GitHub para almacenar el código fuente de todos los servicios. Cada componente debe estar en una carpeta aparte y no debe haber acoplamiento entre ellas.
El video de la entrega debe estar embebido dentro de la descripción del release, no puede estar publicado en otras plataformas ni se puede entregar por medio de enlaces. Si el video no se encuentra en la descripción del release, la nota de este será de cero.

Restricciones de la primera entrega

Usar PostgreSQL como base de datos. Otras bases de datos no deben ser utilizadas a menos que se indique de manera explicita en el enunciado.
    El puerto de todas las base de datos DEBE ser `5432`.
El token de autenticación de usuario debe ser una cadena de caracteres random generada en el código, NO debe utilizar JWT. Puede usar por ejemplo un uuid.
Las fechas deben manejarse en formato ISO `yyyy-mm-ddTHH:MM:SS`. La zona horaria será UTC 0.
Todas las aplicaciones con bases de datos deben exponer un endpoint POST `/reset` que limpie la base de datos.
Todas las aplicaciones con bases de datos deben exponer un endpoint GET `/ping` que permite validar que la aplicación está en ejecución.
El lenguaje de programación es de libre elección del equipo, pero en cualquier lenguaje se debe cumplir con el 70% de cobertura del código. Antes de iniciar el desarrollo debe validar con el equipo de tutores el lenguaje y la librería para verificar la cobertura del código.

Para esta primera entrega se requiere un mínimo de 4GB de RAM y 2 Cores para correr todas las aplicaciones de forma local. Tenga presente estos limitantes para preparar su ambiente de desarrollo.
El namespace de despliegue en K8s debe ser `default`. No use namespaces distintos.

Table of contents

Vista de información

Entidad usuario

Campo	Tipo de dato	Descripción
id	cadena de caracteres	identificador del usuario en formato uuid.
username	cadena de caracteres sin espacios y caracteres especiales	nombre de usuario

email	cadena de caracteres en formato de correo electrónico	dirección de correo electrónico del usuario
phoneNumber	cadena de caracteres (opcional)	número de teléfono para contacto
dni	cadena de caracteres (opcional)	documento de identidad del usuario
fullName	cadena de caracteres (opcional)	nombre completo del usuario
password	cadena de caracteres	password cifrado del usuario
salt	cadena de caracteres	sal para el cifrado del password leer
token	cadena de caracteres	Token actual del usuario

status	cadena de caracteres	Estado del usuario, solo una de estas opciones es válida: POR_VERIFICAR, NO_VERIFICADO, VERIFICADO
expireAt	datetime	fecha y hora de vencimiento de la última sesión (token generado).
createdAt	datetime	fecha y hora de creación del usuario
updatedAt	datetime	fecha y hora de actualización del usuario

Entidad Trayecto

Campo	**Tipo de dato**	**Descripción**
id	cadena de caracteres	identificador del trayecto en formato uuid.
flightId	cadena de caracteres	identificador del vuelo por ejemplo (AA001), este campo debe ser único.
sourceAirportCode	cadena de caracteres	código del aeropuerto de origen (ver)

sourceCountry	cadena de caracteres	país de origen
destinyAirportCode	cadena de caracteres	código del aeropuerto de destino (ver)
destinyCountry	cadena de caracteres	país de destino
bagCost	número entero	costo del envío de una maleta en el trayecto
plannedStartDate	datetime	fecha y hora planeada de inicio del trayecto
plannedEndDate	datetime	fecha y hora planeada de finalización del trayecto
createdAt	datetime	fecha y hora de creación del trayecto
updatedAt	datetime	fecha y hora de actualización del trayecto

Entidad Publicación

Campo	Tipo de dato	Descripción

id	cadena de caracteres	identificador de la publicación en formato uuid.
routeId	cadena de caracteres	identificador del trayecto
userId	cadena de caracteres	identificador del usuario dueño de la publicación
expireAt	datetime	fecha y hora máxima en que se reciben ofertas
createdAt	datetime	fecha y hora de creación de la publicación

Entidad Oferta

Campo	**Tipo de dato**	**Descripción**
id	cadena de caracteres	identificador de la oferta en formato uuid.
postId	cadena de caracteres	identificador de la publicación asociada a la oferta

userId	cadena de caracteres	identificador del usuario que realizó la oferta
description	cadena de caracteres	descripción de no más de 140 caracteres sobre el paquete a llevar.
size	cadena de caracteres	un valor que describe subjetivamente del tamaño del paquete, puede ser LARGE, MEDIUM, SMALL
fragile	booleano	si es un paquete delicado o no
offer	número	valor en dólares de la oferta por llevar el paquete
createdAt	datetime	fecha y hora de creación de la publicación

> Omita la entidad Score por el momento.

Vista funcional

Este modelo presenta cuales son algunos de los componentes en ejecución que se han definido para la arquitectura. En este no se presentan todos los que se espera que tenga el sistema, sino aquellos necesarios para la primera iteración del proyecto.

Cada componente de ejecución expone un API REST, la documentación la encuentra relacionada en la siguiente tabla.

Table of contents

API Usuarios

El API de usuarios permite la gestión de usuarios con funcionalidades como crear usuarios y validar la identidad de un usuario por medio de tokens.

1. Creación de usuarios

Crea un usuario con los datos brindados, el nombre del usuario debe ser único, así como el correo.

Método	POST
Ruta	/users
Parámetros	

Encabezados	

Cuerpo	{
	"username": nombre de usuario,
	"password": contraseña del usuario,
	"email": correo electrónico del usuario,
	"dni": identificación,
	"fullName": nombre completo del usuario,
	"phoneNumber": número de teléfono
	}

Respuestas

Código	Descripción	Cuerpo
400	En el caso que el campo username, password o email, no esté presente en la solicitud, o los valores enviados no correspondan a lo esperado.	
412	En el caso que el usuario con el username o el correo ya exista.	

201	En el caso que el usuario se haya creado con éxito.	{
		"id": id del usuario,
		"createdAt": fecha y hora de creación del usuario en formato ISO
		}

2. Actualización de usuarios

Actualiza los datos de un usuario con los datos brindados, solo los valores de fullName, phoneNumber, dni, status. Todos los valores son opcionales, solo se modifican los que se reciben, los demás permanecen sin modificarse. Este endpoint será público por decisión de arquitectura.

Método	PATCH
Ruta	/users/{id}
Parámetros	id: identificador del usuario
Encabezados	

Cuerpo	{
	"status": nuevo estado del usuario,
	"dni": identificación,
	"fullName": nombre completo del usuario,
	"phoneNumber": número de teléfono
	}

Respuestas

Código	Descripción	Cuerpo
400	Si la petición no contiene al menos uno de los campos esperados.	
404	En el caso que el usuario con id no exista.	

200	En el caso que el usuario se haya actualizado con éxito.	{
		"msg": "el usuario ha sido actualizado"
		}

3. Generación de token

Genera un nuevo Token para el usuario correspondiente al username y contraseña.

Método	POST
Ruta	/users/auth
Parámetros	

Encabezados	

Cuerpo	{
	"username": nombre de usuario,
	"password": contraseña del usuario
	}

Respuestas

Código	Descripción	Cuerpo
400	En el caso que alguno de los campos no esté presente en la solicitud.	
404	En el caso que el usuario con username y password no exista.	

200	Si las credenciales son válidas.	{
		"id": id del usuario,
		"token": Token generado para la sesión del usuario,
		"expireAt": fecha y hora de vencimiento del token en formato ISO
		}

4. Consultar información del usuario

Retorna los datos del usuario al que pertenece el token.

Método	GET
Ruta	/users/me
Parámetros	
Encabezados	`Authorization: Bearer token`
Cuerpo	

Respuestas

Código	Descripción	Cuerpo

401	El token no es válido o está vencido.	
403	El token no está en el encabezado de la solicitud.	

200	En el caso que el token sea válido.	{
		"id": identificador del usuario,
		"username": nombre de usuario,
		"email": correo electrónico del usuario,
		"fullName": nombre completo del usuario,
		"dni": número de identificación del usuario,
		"phoneNumber": número de contacto telefónico del usuario,
		"status": Estado del usuario en el sistema
		}

5. Consultar cantidad de entidades

Usado para consultar cuantas entidades de usuarios están en la base de datos

Método	GET
Ruta	/users/count
Parámetros	

Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Cantidad de entidades almacenadas en la base de datos.	`{"count": número positivo o cero}`

6. Consulta de salud del servicio

Usado para verificar el estado del servicio.

Método	GET
Ruta	/users/ping
Parámetros	
Encabezados	

Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Solo para confirmar que el servicio está arriba.	pong

7. Restablecer base de datos

Usado para borrar todos los datos de usuarios.

Método	POST
Ruta	/users/reset
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo

200	Todos los datos fueron eliminados.	{
		"msg": "Todos los datos fueron eliminados"
		}

API Trayectos

El API de trayectos permite crear trayectos (rutas) para ser usados por las publicaciones.

1. Creación de trayecto

Crea un trayecto con los datos brindados, solo un usuario autorizado puede realizar esta operación.

Método	POST
Ruta	/routes
Parámetros	
Encabezados	

Cuerpo	{
	"flightId": código del vuelo,
	"sourceAirportCode": código del aeropuerto de origen,
	"sourceCountry": nombre del país de origen,
	"destinyAirportCode": código del aeropuerto de destino,
	"destinyCountry": nombre del país de destino,
	"bagCost": costo de envío de maleta,
	"plannedStartDate": fecha y hora de inicio del travecto,
	"plannedEndDate": fecha y hora de finalización del travecto
	}

Respuestas

Código	Descripción	Cuerpo
400	En el caso que alguno de los campos no esté presente en la solicitud.	
412	En el caso que el flightId ya existe.	

412	En el caso que la fecha de inicio y fin del trayecto no sean válidas; fechas en el pasado o no consecutivas.	`{ "msg": "Las fechas del trayecto no son válidas" }`
201	En el caso que el trayecto se haya creado con éxito.	`{ "id": id del trayecto, "createdAt": fecha y hora de creación del trayecto en formato ISO }`

2. Ver y filtrar trayectos

Retorna todos los trayectos o aquellos que corresponden a los parámetros de búsqueda. Solo un usuario autorizado puede realizar esta operación.

Método	GET
Ruta	/routes?flight={flightId}
Parámetros	flightId: id del vuelo, este campo es opcional
Encabezados	

Cuerpo	

Respuestas

Código	Descripción	Cuerpo
400	En el caso que alguno de los campos de búsqueda no tenga el formato esperado.	

200	Listado de trayectos.	[
		{
		"id": id del trayecto
		"flightId": código del vuelo,
		"sourceAirportCode": código del aeropuerto de origen,
		"sourceCountry": nombre del país de origen,
		"destinyAirportCode": código del aeropuerto de destino,
		"destinyCountry": nombre del país de destino,
		"bagCost": costo de envío de maleta,
		"plannedStartDate": fecha y hora de inicio del trayecto,
		"plannedEndDate": fecha y hora de finalización del trayecto,
		"createdAt": fecha y hora de creación del trayecto en formato ISO
		}
		]

3. Consultar un trayecto

Retorna un trayecto, solo un usuario autorizado puede realizar esta operación.

Método	GET
Ruta	/routes/{id}
Parámetros	id: identificador del trayecto
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
400	El id no es un valor string con formato uuid.	
404	El trayecto con ese id no existe.	

200	Trayecto que corresponde al identificador.	{
		"id": id del trayecto
		"flightId": código del vuelo,
		"sourceAirportCode": código del aeropuerto de origen,
		"sourceCountry": nombre del país de origen,
		"destinyAirportCode": código del aeropuerto de destino,
		"destinyCountry": nombre del país de destino,
		"bagCost": costo de envío de maleta,
		"plannedStartDate": fecha y hora de inicio del trayecto,
		"plannedEndDate": fecha y hora de finalización del trayecto,
		"createdAt": fecha y hora de creación del trayecto en formato ISO
		}

4. Eliminar trayecto

Elimina el trayecto, solo un usuario autorizado puede realizar esta operación.

Método	DELETE
Ruta	/routes/{id}
Parámetros	id: identificador del trayecto
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
400	El id no es un valor string con formato uuid.	
404	El trayecto con ese id no existe.	

200	El trayecto fue eliminado.	{
		"msg": "el trayecto fue eliminado"
		}

5. Consultar cantidad de entidades

Usado para consultar cuantas entidades de rutas están en la base de datos.

Método	GET
Ruta	/routes/count
Parámetros	
Encabezados	

Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Cantidad de entidades almacenadas en la base de datos.	{"count": número positivo o cero}

6. Consulta de salud del servicio

Usado para verificar el estado del servicio.

Método	GET
Ruta	/routes/ping
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Solo para confirmar que el servicio está arriba.	pong

7. Restablecer base de datos

Usado para borrar todos los datos de rutas.

Método	POST
Ruta	/routes/reset
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo

200	Todos los datos fueron eliminados.	{
		"msg": "Todos los datos fueron eliminados"
		}

API Publicaciones

El API de publicaciones permite crear, buscar, eliminar y consultar publicaciones.

1. Creación de publicación

Crea una publicación en el sistema.

Método	POST
Ruta	/posts
Parámetros	
Encabezados	

Cuerpo	{
	"routeId": id del trayecto,
	"expireAt": fecha y hora máxima en que se recibirán ofertas en formato ISO,
	"userId": id del usuario dueño de la publicación
	}

Respuestas

Código	Descripción	Cuerpo
400	En el caso que alguno de los campos no esté presente en la solicitud, o no tengan el formato esperado.	

412	En el caso que la fecha de expiración no sea en el futuro o no sea válida.	{
		"msg": "La fecha expiración no es válida"
		}

201	En el caso que la publicación se haya creado con éxito.	{
		"id": id de la publicación,
		"userId": id del usuario que creo la publicación,
		"createdAt": fecha y hora de creación de la publicación en formato ISO
		}

2. Ver y filtrar publicaciones

Retorna el listado de publicaciones que coinciden con los parámetros brindados.

Método	GET
Ruta	/posts?expire={true|false}&route={routeId}&owner={userId}
Parámetros	Todos los parámetros son opcionales, y su funcionamiento es de tipo AND. expire: filtra por publicaciones expiradas o no expiradas route: id del trayecto que se desea usar para el envío. owner: dueño de la publicación. En el caso de que ninguno esté presente se devolverá la lista de datos sin filtrar. Es decir, todas las publicaciones.
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
400	En el caso que alguno de los campos de búsqueda no tenga el formato esperado.	

200	Listado de publicaciones que corresponden a la búsqueda.	[
		{
		"id": id de la publicación
		"routeId": id del trayecto,
		"userId": id del usuario owner de la publicación,
		"expireAt": fecha y hora máxima en que se recibirán ofertas en formato ISO,
		"createdAt": fecha y hora de creación de la publicación en formato ISO
		}
		]

3. Consultar una publicación

Retorna una publicación con el id en la solicitud.

Método	GET
Ruta	/posts/{id}
Parámetros	id: identificador de la publicación

Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
400	El id no es un valor string con formato uuid.	
404	La publicación con ese id no existe.	
200	Publicación que corresponde al identificador.	{ "id": id de la publicación "routeId": id del trayecto, "userId": id del usuario dueño de la publicación, "expireAt": fecha y hora máxima en que se recibirán ofertas en formato ISO, "createdAt": fecha y hora de creación de la publicación en formato ISO }

4. Eliminar publicación

Elimina una publicación.

Método	DELETE
Ruta	/posts/{id}
Parámetros	id: identificador de la publicación.
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
400	El id no es un valor string con formato uuid.	
404	La publicación con ese id no existe.	

200	La publicación fue eliminada.	{
		"msg": "la publicación fue eliminada"
		}

5. Consultar cantidad de entidades

Usado para consultar cuantas entidades de publicaciones están en la base de datos.

Método	GET
Ruta	/posts/count
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo

200	Cantidad de entidades almacenadas en la base de datos.	`{"count": número positivo o cero}`

6. Consulta de salud del servicio

Usado para verificar el estado del servicio.

Método	GET
Ruta	/posts/ping
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Solo para confirmar que el servicio está arriba.	`pong`

7. Restablecer base de datos

Usado para borrar todas las publicaciones.

Método	POST
Ruta	/posts/reset
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Todos los datos fueron eliminados.	{"msg": "Todos los datos fueron eliminados"}

API Ofertas

El API de ofertas permite crear, buscar, eliminar y consultar ofertas.

1. Creación de ofertas

Crea una oferta asociada.

Método	POST
Ruta	/offers
Parámetros	
Encabezados	
Cuerpo	{ "postId": id de la publicación, "userId": id del usuario dueño de la oferta, "description": descripción del paquete a llevar, "size": LARGE|MEDIUM|SMALL "fragile" : booleano que indica si es un paquete delicado o no, "offer": valor en dólares de la oferta para llevar el paquete }

Respuestas

Código	Descripción	Cuerpo

400	En el caso que alguno de los campos no esté presente en la solicitud, o no tengan el formato esperado.	
412	En el caso que los valores no estén entre lo esperado, por ejemplo el tamaño del paquete no sea válido o la oferta sea negativa.	
201	En el caso que la oferta se haya creado con éxito.	{ "id": id de la oferta, "userId": id del usuario que creo la oferta, "createdAt": fecha y hora de creación de la oferta en formato ISO }

2. Ver y filtrar ofertas

Retorna el listado de ofertas que coinciden con los parámetros brindados.

Método	GET
Ruta	/offers?post={postId}&owner={userId}
Parámetros	Todos los parámetros son opcionales, y su funcionamiento es de tipo AND. post: id de la publicación que se desea usar para el envío. owner: dueño de la oferta. En el caso de que ninguno esté presente se devolverá la lista de datos sin filtrar. Es decir, todas las ofertas.
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
400	En el caso que alguno de los campos de búsqueda no tenga el formato esperado.	

200	Listado de ofertas que corresponden a la búsqueda.	[
		  {
		  "id": id de la oferta
		  "postId": id de la publicación,
		  "description": descripción del paquete a llevar,
		  "size": LARGE ó MEDIUM ó SMALL,
		  "fragile" : booleano que indica si es un paquete delicado o no,
		  "offer": valor en dólares de la oferta para llevar el paquete,
		  "createdAt": fecha de creación de la oferta en formato ISO,
		  "userId": id del usuario que creo la oferta
		  }
		]

3. Consultar una oferta

Retorna una oferta.

Método	GET
Ruta	/offers/{id}
Parámetros	id: identificador de la oferta
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
400	El id no es un valor string con formato uuid.	
404	La oferta con ese id no existe.	
200	Oferta que corresponde al identificador.	{ "id": id de la oferta "postId": id de la publicación, "description": descripción del paquete a llevar, "size": LARGE ó MEDIUM ó SMALL, "fragile" : booleano que indica si es un paquete delicado o no, "offer": valor en dólares de la oferta para llevar el paquete, "createdAt": fecha de creación de la oferta en formato ISO, "userId": id del usuario que creo la oferta, }

4. Eliminar oferta

Elimina una oferta.

Método	DELETE
Ruta	/offers/{id}

Parámetros	id: identificador de la oferta.
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
400	El id no es un valor string con formato uuid.	
404	La oferta con ese id no existe.	

200	La oferta fue eliminada.	{
		"msg": "la oferta fue eliminada"
		}

5. Consultar cantidad de entidades

Usado para consultar cuantas entidades de ofertas están en la base de datos

Método	GET

Ruta	/offers/count
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Cantidad de entidades almacenadas en la base de datos.	`{"count": número positivo o cero}`

6. Consulta de salud del servicio

Usado para verificar el estado del servicio.

Método	GET
Ruta	/offers/ping
Parámetros	

Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Solo para confirmar que el servicio está arriba.	`pong`

7. Restablecer base de datos

Usado para eliminar todas las ofertas.

Método	POST
Ruta	/offers/reset
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Todos los datos fueron eliminados.	{"msg": "Todos los datos fueron eliminados"}

Vista de despliegue

En este documento se presentan los diferentes diagramas que indican como se deben desplegar los componentes definidos en la arquitectura.

Modelo de despliegue (Local)

De manera local el sistema será desplegado por medio de contenedores haciendo uso de docker, cada componente se ejecutará en un contenedor independiente, al igual que sus bases de datos.

La comunicación con las aplicaciones será por medio del protocolo *HTTP* haciendo uso del API REST definido en la vista funcional.

En la primera entrega las aplicaciones están completamente desacopladas y no tienen comunicación entre ellas. Además las bases de datos se ejecutan usando el mismo modelo que las aplicaciones, algo que cambiará para futuras entregas.

Tenga presente que las bases de datos deben configurarse con volúmenes.

Modelo de red (Local)

Cada aplicación tiene una red definida con su base de datos. De esta forma el sistema restringe que una aplicación pueda acceder a los datos de otra y garantiza el aislamiento y bajo acoplamiento entre aplicaciones.

Tambien se observa en el modelo que los servicios de base de datos son tipo ClusterIP dado que solo permitirán comunicación interna, mientras los servicios de las aplicaciones son tipo NodePort con el fin de permitir la comunicación por fuera del cluster.

Vista de despliegue

En este documento se presentan los diferentes diagramas que indican como se deben desplegar los componentes definidos en la arquitectura.

Modelo de despliegue (Local)

De manera local el sistema será desplegado por medio de contenedores haciendo uso de docker, cada componente se ejecutará en un contenedor independiente, al igual que sus bases de datos.

La comunicación con las aplicaciones será por medio del protocolo *HTTP* haciendo uso del API REST definido en la vista funcional.

En la primera entrega las aplicaciones están completamente desacopladas y no tienen comunicación entre ellas. Además las bases de datos se ejecutan usando el mismo modelo que las aplicaciones, algo que cambiará para futuras entregas.

Tenga presente que las bases de datos deben configurarse con volúmenes.

Modelo de red (Local)

Cada aplicación tiene una red definida con su base de datos. De esta forma el sistema restringe que una aplicación pueda acceder a los datos de otra y garantiza el aislamiento y bajo acoplamiento entre aplicaciones.

Tambien se observa en el modelo que los servicios de base de datos son tipo ClusterIP dado que solo permitirán comunicación interna, mientras los servicios de las aplicaciones son tipo NodePort con el fin de permitir la comunicación por fuera del cluster.

Recursos para la entrega 1

Código

> El código que encuentra en los repositorios de ejemplo siguen una estructura y arquitectura que es OPCIONAL. Usted junto con su equipo pueden definir si usan esa arquitectura u otra, pero tenga presente que se espera que en cada carpeta de aplicación exista el archivo Dockerfile.

Su equipo cuenta con un repositorio base de código que contempla:

- Estructura básica del proyecto.
- Aplicación de ejemplo con `Python 3.11`, `Poetry`, `FastAPI`, `Docker`, `Pytest` y `Vale`.

Archivo Makefile el cuál contiene reglas que no pueden ser modificadas, pero si puede agregar nuevas reglas.
Pipelines de github para evaluar su entrega (`.github/workflows/ci_evaluador_entrega1_k8s.yml`, `.github/workflows/ci_evaluador_entrega1_docs.yml`). Ninguno de estos archivos debe ser manipulado.
Código de ejemplo de configuración de pipeline de github para pruebas unitarias (`.github/workflows/ci_evaluador_unit.yml`) y que verifica 70% de la cobertura de código (Pestaña Actions). No cree otros archivos, agregue un job por cada aplicación a probar.
Ejemplo de pruebas unitarias en `Python`.

Si no desea trabajar con `FastAPI` y `Poetry`, puede usar este ejemplo de aplicación que hace uso de `Pipfile` y `Flask`: Ejemplo flask.
Ejemplos de `Python` para usar `Pytest` y mocks de `requests` https://github.com/MISW-4301-Desarrollo-Apps-en-la-Nube/ejemplos_python.

Pruebas

Para probar sus aplicaciones localmente y validar que responden correctamente a la especificación de API REST, puede usar las siguientes colecciones. Descargue cada archivo JSON e impórtelo en su aplicación de Postman. Estas mismas pruebas son usadas para la calificación de su entrega.

Pruebas para Users:
https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/proyecto-monitor/main/entrega1/entrega1_users.json
Pruebas para Posts:
https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/proyecto-monitor/main/entrega1/entrega1_posts.json
Pruebas para Offers:
https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/proyecto-monitor/main/entrega1/entrega1_offers.json
Pruebas para Routes:
https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/proyecto-monitor/main/entrega1/entrega1_routes.json

Para importar la colección use el botón Import en la sección superior izquierda.

Una vez importada la colección, actualice la variable de ambiente de la ruta de la aplicación que desea probar. No modifique el `Initial value` (valor inicial), modifique el `Current value` (valor actual).

Finalmente, ejecute la colección haciendo clic derecho en el nombre de la colección y seleccionando `Run`. Si la ruta de la aplicación está correctamente configurada Postman ejecutará múltiples solicitudes de API y también ejecutará algunos assertions que hemos preparado para asegurarnos de que la aplicación esté funcionando como se espera.

Pipelines

En su repositorio de equipo encontrará una serie de pipelines para verificar que su entrega cumple con parte de los entregables calificables. Estos pipelines se ejecutan como un workflow de Github Actions de forma manual o automática dependiendo del workflow.

El workflow que revisa las pruebas unitarias se ejecuta cada que usted mezcle en la rama `main` o cuando haga un Pull Request a la rama `main`. El workflow que verifica la documentación `Evaluador Documentación Entrega 1` y el que ejecuta las pruebas de API se deben ejecutar manualmente y los puede ejecutar en cualquier rama. Para ello diríjase a la sección de `Actions` del repositorio.

Encontrará en el menú izquierdo una lista de todos los flujos de trabajo (workflows) disponibles para ejecución. En este caso, verá `Evaluador Documentación Entrega 1` y `Evaluador Implementación Entrega 1` que corresponden al pipeline documentación y al de pruebas de sus aplicaciones sobre su despliegue en k8s respectivamente. Haga clic en el workflow y verá un botón `Run workflow` en la sección superior derecha. Haga clic en este botón, seleccione la rama en la que desea ejecutarlo y por último oprima el botón `Run workflow`.

Esto iniciará la ejecución del workflow en la rama. Si todo funciona correctamente y la entrega es correcta, verá que todas las comprobaciones aparecen como aprobadas (passed).

Tecnologías del proyecto

El equipo de desarrollo podrá seleccionar los lenguajes con los que desarrollarán las aplicaciones, sin embargo, las tecnologías de ejecución y despliegue, así como las tecnologías de bases de datos, no serán opcionales y solo podrán usarse las ya definidas. Le recomendamos realizar la selección de lenguajes durante la primera reunión de equipo.

Tenga presente que en los ejemplos y tutoriales se hace uso de Python como lenguaje de programación, y los tutores están capacitados para resolver dudas de implementación en este lenguaje. La selección de otro lenguaje no garantiza que el equipo de tutores le pueda colaborar.

En caso de tener dudas sobre si puede hacer uso de X o Y tecnología, favor haga uso del canal de Slack para comunicar su duda.

Tecnologías requeridas desde la primera entrega

- Postman
- Git. No olvide configurar su usuario de git para que las entregas estén marcadas con este.
- Python 3.11. Para ejecución de ejemplos y en el caso decida usar este lenguaje en su proyecto.
    - Poetry. Herramienta recomendada para la gestión de dependencias y manejo del ambiente virtual.
    - Pyenv. Herramienta para gestionar diferentes versiones de python en su ambiente.
- IDE de su elección. Puede definirlo con su equipo de trabajo.
- Docker
- Minikube. Para ejecución local de kubernetes.
- Kubectl. Cliente de kubernetes.

Entrega 2

Comprende los temas vistos hasta la semana 4 y su fecha límite es el sábado de la semana 6 11:59 pm GMT - 5.

Table of contents

Enunciado de la entrega 2

TABLE OF CONTENTS

Objetivos

Evaluar la capacidad de los estudiantes para diseñar una solución basada en microservicios, considerando un conjunto de restricciones tecnológicas definidas.
Implementar y desplegar un sistema compuesto por aplicaciones contenerizadas, utilizando servicios de orquestación y almacenamiento

provistos por un proveedor de nube, y aplicando buenas prácticas de desarrollo y calidad de código.

El objetivo de su equipo en esta entrega es documentar, diseñar, implementar y desplegar los componentes necesarios para cumplir con una serie de requerimientos. Cabe resaltar que debe hacer uso del conjunto de servicios ya implementados y los patrones de diseño vistos durante estas semanas.

Prerrequisitos y restricciones

La entrega debe cumplir con las restricciones descritas en el documento de restricciones y utilizar las tecnologías listadas en el documento de tecnologías.

Para esta entrega es necesario que los componentes realizados en la entrega 1 estén completos y funcionales. Si su equipo no logró terminar esa entrega a tiempo y aún presenta errores, tiene dos opciones:

> Terminar la implementación durante la primera semana de esta entrega.
> La dirección del curso le entregará un proyecto ya solucionado que tiene los 4 microservicios ya probados y funcionando. Sin embargo, tenga presente que NO incluye ni capacitación, ni documentación adicional al archivo `README.md`, ni soporte, ni pruebas unitarias.

Alcance

El equipo fundador se ha reunido con el equipo de ingeniería para ultimar detalles sobre los requerimientos funcionales que la primera versión de la aplicación debe ofrecer. La documentación de esos requerimientos se encuentra en la sección 'Requerimientos Funcionales'.

Al revisar los requerimientos considere que el RF-001 y RF-002 ya están cubiertos parcialmente por la implementación actual (la primera entrega). Por lo tanto, para esta

segunda entrega su equipo debe enfocar los esfuerzos en los otros requerimientos documentados:

- RF-003
- RF-004
- RF-005

> Favor lea los criterios de aceptación de cada uno de los requerimientos para tener claridad de como se calificará la entrega.

En los nuevos requerimientos se introduce una nueva entidad en el sistema: Utilidad (Score). Por lo tanto, se debe diseñar e implementar un nuevo servicio responsable de gestionar la información relacionada con esta entidad.

Tenga en cuenta los siguientes aspectos al diseñar la aplicación de scores:

- Tanto el diseño del API como el esquema de datos correspondiente, serán definidos por su equipo.
- El valor de Score debe ser obtenido desde otro servicio, y puede no ser consistente en el tiempo. En consecuencia, su implementación debe ser resiliente ante escenarios en los que una oferta no tenga un Score definido.
- Esta no es una aplicación de cálculo únicamente, sus datos deben ser persistidos en una base de datos dado que el Score podría cambiar en el tiempo.

> El concepto de utilidad (score) y como se calcula lo puede encontrar en el glosario del proyecto.

Cada requerimiento debe cumplir con un diseño de API ya definido. La documentación de cada uno la puede encontrar en la siguiente tabla:

Requerimiento	**Descripción**	**Link**
RF-003	Crear publicaciones	Link
RF-004	Crear ofertas	Link

RF-005	Consulta de publicaciones	Link

Todos los componentes deben ser desplegados sobre un proveedor de nube haciendo uso de los servicios de bases de datos y orquestación de contenedores ya definido.

Los entregables de esta entrega son:

Documentación

Haciendo uso de la información de los requerimientos y la descripción de los servicios ya implementados en el sistema (entrega 1), el equipo de trabajo debe construir un documento de arquitectura donde se describen los microservicios y decisiones de diseño que solucionan los tres requerimientos de esta entrega. Esta documentación debe estar almacenada y debe ser navegable en la carpeta `/docs`.

El diseño debe contemplar las siguientes restricciones de diseño.
La solución debe contemplar que el usuario final solo conocerá UN punto de acceso al sistema (un solo host) y por lo tanto hará UNA sola petición HTTP por cada requerimiento, y no una secuencia. El sistema recibirá cada solicitud y se encargará de que todos los componentes trabajen juntos para completar cada objetivo.
La solución debe contemplar una alta demanda de usuarios que se incrementará con el tiempo.
La solución debe contemplar que solo usuarios autenticados y autorizados pueden hacer uso de estos nuevos servicios.
La solución debe contemplar todo lo que se encuentra implementado. No se deben duplicar funcionalidades.
La solución debe contemplar que trabajamos con microservicios y no debemos tener acoplamientos que nos impacten la escalabilidad, disponibilidad, facilidad de desecho y mantenibilidad de nuestras aplicaciones. - Los enlaces a documentación externa están totalmente prohibidos, toda la documentación y modelos deben estar contenidos en la misma carpeta, incluyendo las imágenes que deben ser creados usando PlantUML. - La documentación debe contener las siguientes elementos:
Un modelo de componentes de la vista funcional en el que se describen los componentes necesarios (tanto nuevos como anteriormente

implementados) para cumplir con los requerimientos. Cada componente NUEVO debe estar documentado usando la siguiente tabla:

Componente	Nombre del componente
Código/Id del componente	Debe aparecer en el diagrama
Tipo	Tipo de componente: base de datos, servicio, almacenamiento, etc
Responsabilidad	Responsabilidad del componente dentro del sistema/aplicación
Consideraciones de diseño	Tradeoffs, impacto en el sistema actual, puntos de dolor, etc.
Integraciones	Con cuáles componentes se comunica, protocolos, verbos, tipo de comunicación

Un modelo de despliegue de la vista de despliegue en el que se describen como los componentes funcionales serán desplegados haciendo uso de los servicios de la nube.
Un módelo de secuencia por cada requerimiento donde se describa el proceso que soluciona el requerimiento. Este debe tener en cuenta flujos de excepción y manejo de errores
Una página donde se describen los patrones con los que se resuelve cada requerimiento. Las tablas deben tener este formato:

Requerimiento	Identificador del requerimiento.
Patrón utilizado	Nombre del patrón utilizado.
Justificación	Justificación de por que este patrón puede ser utilizado y para qué se debe usar.
Atributos de calidad favorecidos	Lista de atributos de calidad que se favorecen al utilizar este patrón en este contexto.
Atributos de calidad desfavorecidos	Lista de atributos de calidad que se desfavorecen al utilizar este patrón en este contexto.
Componentes involucrados	Lista de los componentes involucrados del modelo para resolver el requerimiento (nombre, código).

La nueva documentación debe estar publicada en su página web y debe ser fácilmente navegable por medio de la tabla de contenido. Por ejemplo:

```
# Documentación técnica

...

 * [Vista de componentes](ruta a la página con la documentación).

 * [Vista de despliegue](ruta a la página con la documentación).

 * [Vista de desarrollo](ruta a la página con la documentación).
```

    * [Descripción de nuevos componentes](ruta a la página con la documentación).

  * Requerimientos

    * Requerimiento RF-003

      * [Documentación patrón solución requerimiento 3](ruta a la página con la documentación).

      * [Explicación del proceso paso a paso de la solución requerimiento 3](ruta a la página con la documentación).

    * Requerimiento RF-004

      * [Documentación patrón solución requerimiento 4](ruta a la página con la documentación).

      * [Explicación del proceso paso a paso de la solución requerimiento 4](ruta a la página con la documentación).

    * Requerimiento RF-005

      * [Documentación patrón solución requerimiento 5](ruta a la página con la documentación).

      * [Explicación del proceso paso a paso de la solución requerimiento 5](ruta a la página con la documentación).

> Actualicen el archivo `README.md` en la raíz del proyecto que indique la estructura de las carpetas y cómo hacer el despliegue.
> Actualicen el archivo `config.yaml` con la siguiente estructura. (Este archivo es necesario para las pruebas por parte del equipo calificador y su entrega es obligatoria).

```
team:
  name: <nombre del equipo>
  members:
    - user: <usuario uniandes>
      percentage: <0 a 100>
    - user: <usuario uniandes>
```

```
        percentage: <0 a 100>
      - user: <usuario uniandes>
        percentage: <0 a 100>
      - user: <usuario uniandes>
        percentage: <0 a 100>
  board: <link al dashboard de tareas del equipo>
  docs: <link al github pages del equipo>
  url: <url del host de la aplicación, ejemplo https://host>
  cluster_name: <Nombre del cluster de K8s>
  users_app:
    ...
  posts_app:
    ...
  offers_app:
    ...
  routes_app:
    ...
  scores_app:
    folder: <nombre de la carpeta donde está la aplicación de scores>
    image_name: <nombre de la imagen de la aplicación de scores, mismo que el nombre
en el archivo de despliegue de k8s>
    image_tag: <tag de la imagen de la aplicación de scores, por ejemplo: latest,
v1.0.0. Debe ser el mismo que el tag en el archivo de despliegue de k8s>
    authors:
      - <usuario uniandes>
```

```
  rf00x:
    folder: <nombre de la carpeta donde se soluciona el requerimiento x>
    image_name: <nombre de la imagen de la aplicación de ofertas, mismo que el nombre en el archivo de despliegue de k8s>
    image_tag: <tag de la imagen de la aplicación de ofertas, por ejemplo: latest, v1.0.0. Debe ser el mismo que el tag en el archivo de despliegue de k8s>
    authors:
      - <usuario uniandes>
  ... # otros componentes (Todos los componentes deben estar listados acá).
```

El archivo `config.yaml` es el archivo más importante en el repositorio. Este archivo contiene la configuración que se usa en los pipelines para evaluar su entrega y se define la calificación de cada miembro del equipo. Tenga en cuenta que, si este archivo no está correctamente configurado, no será posible calificar la entrega.

Formato de entrega

> Archivos markdown en el repositorio del equipo en el release `ProyectoSegundaEntrega`.
> Página web desplegada en github pages y creada a partir de los archivos en la carpeta `/docs`.

Este entregable se califica sobre un release que debe ser creado en el repositorio de su equipo y que debe llevar el nombre `ProyectoSegundaEntrega`. El release debe ser creado antes de la hora límite de entrega del proyecto. Si el release es creado posterior a la hora límite, se aplicarán las sanciones descritas en la sección de calificación del curso. La calificación se realizará con base en el contenido del release marcado como `ProyectoSegundaEntrega`, por lo que cualquier cambio fuera de este no será considerado.

Mecanismo de evaluación

Este entregable se evaluará de dos formas:

- Manualmente, los tutores del curso revisarán a detalle su página web para validar que todas las unidades y los modelos son correctos. Tenga presente que la página web de su proyecto debe ser privada.

Implementación

Siguiendo el diseño documentado, como equipo deben realizar la implementación funcional de los componentes documentados.

Importante: los microservicios implementados en la primera entrega NO deben modificarse. En caso de dudas revise el documento de restricciones.

- Código funcional de todas las aplicaciones del sistema siguiendo la documentación creada por el equipo.
  - Se seguirá usando el mismo repositorio de código de Github de la primera entrega. Todo el código debe quedar sobre la rama master o main, no se calificará en otras ramas.
  - Cada microservicio puede ser construido en el lenguaje de su preferencia; Python, JavasCript, Java, etc. Valide con su tutor la tecnología que desea usar en el caso que no use la recomendada por el curso (Python).
  - Continue con la misma estructura de carpetas que trabajó en la entrega anterior, en el que cada aplicación debe estar almacenada en su propia carpeta.
  - Cada componente está formado por el código fuente, sus pruebas unitarias, un archivo `README.md` con la descripción del componente y el archivo Dockerfile.
- Las aplicaciones deben debe ser desplegadas en contenedores haciendo uso de Docker y orquestadas por medio de Kubernetes sobre un proveedor de nube.
  - Se hará uso de UNA sola base de datos sobre el proveedor de nube seleccionado en el curso, la cuál albergará todas las tablas desacopladas. La base de datos debe ser Postgres.
  - Todos los componentes deben ser desplegados sobre el servicio de orquestación del proveedor de nube seleccionado, NO sobre un ambiente local.
  - En la carpeta `k8s` se deben encontrar los archivos necesarios para el despliegue de todos los componentes de Kubernetes. Documente la estructura de sus archivos en la vista de desarrollo del proyecto.

Por medio de la URL de su sistema se debe poder acceder a los endpoints que procesan los requerimientos del enunciado, y a los endpoints de los servicios de la entrega 1. Se validarán que TODOS los endpoints estén presentes, sin excepción. La estructura de endpoints debe ser la siguiente:

`https://<host>/posts/*`

`https://<host>/offers/*`

`https://<host>/routes/*`

`https://<host>/users/*`

`https://<host>/rf003/posts`

`https://<host>/rf004/posts/{id}/offers`

`https://<host>/rf005/posts/{id}`

Formato de entrega

Link del release `ProyectoSegundaEntrega` creado antes de la fecha y hora máxima de entrega.

Este entregable se califica sobre un release que debe ser creado en el repositorio de su equipo y que debe llevar el nombre `ProyectoSegundaEntrega`. El release debe ser creado antes de la hora límite de entrega del proyecto. Si el release es creado posterior a la hora límite, se aplicarán las sanciones descritas en la sección de calificación del curso. La calificación se realizará con base en el contenido del release marcado como `ProyectoSegundaEntrega`, por lo que cualquier cambio fuera de este no será considerado.

Mecanismo de evaluación

Este entregable se evaluará de dos formas:

Por medio de pipelines. En su repositorio encontrará el pipeline `ci_evaluador_entrega2.yml`, el cuál debe ejecutarse de manera exitosa sobre el último commit de su release. Si este archivo es modificado el entregable será penitenciado como indica la rúbrica. El pipeline realiza las siguientes tareas:

Ejecuta pruebas de API sobre cada aplicación para validar que la aplicación si responde al API definido.

Bloquea tráfico entre aplicaciones para validar la consistencia del sistema.

Manualmente se verificará que la solución entregada sea acorde a las decisiones de diseño planteadas en el enunciado: Componentes, despliegue, estructura. Se evaluará la cohesión y aislamiento; que una aplicación no gestione o tenga acceso a información que debe ser gestionada por otras aplicaciones.

Pruebas

Las pruebas unitarias de las aplicaciones de la primera entrega deben seguir funcionando en la ejecución del pipeline y cubrir el 70% de código.

El código debe contar con pruebas unitarias que cubran al menos 70% del código de cada aplicación. La ejecución de las pruebas se validará únicamente en pipelines. Si las pruebas no son ejecutadas y la cobertura no es verificada por el pipeline el entregable será penitenciado como indica la rúbrica. NO se admiten ejecuciones locales.
Se debe modificar el archivo de pipelines con el nombre `ci_evaluador_unit.yml` para que contenga todos los jobs de pruebas unitarias. Cada job debe estar diseñado para probar el cubrimiento de una aplicación, y este no debe ser menor al 70% (En el código fuente del repositorio de su equipo encontrará un ejemplo de un proyecto en Python y como está configurado un pipeline de github).
Las pruebas unitarias serán opcionales para todas las nuevas aplicaciones agregadas al repositorio. Su implementación no se tomará en cuenta para la calificación de la entrega.

Formato de entrega

Link del release `ProyectoSegundaEntrega` creado antes de la fecha y hora máxima de entrega.

Este entregable se califica sobre un release que debe ser creado en el repositorio de su equipo y que debe llevar el nombre `ProyectoSegundaEntrega`. El release debe ser creado antes de la hora límite de entrega del proyecto. Si el release es creado posterior a la hora límite, se aplicarán las sanciones descritas en la sección de calificación del curso. La calificación se realizará con base en el contenido del release marcado como `ProyectoSegundaEntrega`, por lo que cualquier cambio fuera de este no será considerado.

Mecanismo de evaluación

Este entregable se evaluará únicamente por medio de pipelines del repositorio. El archivo `ci_evaluador_unit.yml` debe contener mínimo 4 jobs (uno por aplicación; posts_app, users_app, offers_app, routes_app) en los que se verifica que el cubrimiento de pruebas es exitoso.

Si el equipo utiliza otro lenguaje diferente a Python, u otro framework de pruebas distinto a pytest, deben agregar una sección en su README describiendo las herramientas que usaron, como funcionan, donde se encuentra la documentación oficial de las tecnologías, y describir ese funcionamiento en su video de sustentación.

Presentación

Grabar un video donde presente su equipo, los detalles más relevantes de su proyecto, y una demo de como funciona el sistema.

> En el video se presenta al equipo que trabajó en la entrega, una descripción del repositorio y un demo de como funciona la aplicación haciendo uso de un cliente como Postman. La duración del video debe estar entre cinco y diez minutos. En caso de exceder el tiempo máximo, se aplicará una penalización en la nota de este entregable.

Formato de entrega

El video debe ser incluido en la descripción de su release `ProyectoSegundaEntrega` en el momento de la creación. El video no puede ser incluido posterior a la creación del release y NO será recibido por ningún otro medio o mecanismo.

Mecanismo de evaluación

El video se evaluará manualmente verificando que describe la estructura del sistema y presenta una demo del funcionamiento de la aplicación. Si el video no cumple con la duración esperada o su calidad no permite evaluar que su contenido es el esperado, el entregable será penitenciado como indica la rúbrica.

Después de la entrega, uno o dos grupos podrán ser seleccionados aleatoriamente por los tutores para realizar una sustentación síncrona en la que deberán exponer su solución y cada uno de los entregables descritos; la nota será calculada con base en esa presentación. Esa sustentación sucederá en los siguientes tres días hábiles a la fecha de entrega.

Calificación

La calificación se hará tomando el éxito de inspecciones manuales y el éxito de los pipelines ejecutados. Tenga presente que el tutor puede, si lo considera necesario, ejecutar las pruebas en otros ambientes y tomar el resultado de esa ejecución como su nota. Será debidamente notificado en el caso que suceda.

La calificación se realizará tomando como base esta rúbrica.

> Después de la entrega, uno o dos grupos podrán ser seleccionados aleatoriamente por los tutores para realizar una sustentación síncrona en la que deberán exponer su solución y cada uno de los entregables descritos; la nota será calculada con base en esa presentación. Esa sustentación sucederá en los siguientes tres días hábiles a la fecha de entrega.

Estrategia recomendada

En esta sección se indica cuál es la estrategia recomendada que usted con su equipo deberían de seguir para cumplir con el entregable. Esta solo es una recomendación, es decisión de su equipo las tareas a desarrollar semana a semana.

División de trabajo

Recomendamos que tres miembros del equipo trabajen cada uno en un requerimiento (por ejemplo un solo miembro que trabaje en la consulta de publicaciones), lo que incluye no solo el código, sino la configuración de la base de datos, los contenedores, sus pruebas y su documentación. Y el cuarto miembro trabaje en la implementación de la aplicación de utilidades (Score) y la solución para la autenticación y autorización de usuarios en el sistema.

¿Qué debería hacer la semana 4?

Revisar del enunciado de la entrega 2.
Planeas las actividades y la división de responsabilidades. Haga uso del tablero Kanban de su repositorio para planear y gestionar su trabajo.
Configurar sus nuevas ramas sobre el mismo repositorio.
Definir como harán uso de las cuentas de AWS para las entregas; cuál será la cuenta de la entrega.
Configurar la base de datos en el proveedor de nube. Recomendamos hacer uso de una sola base de datos con múltiples tablas no relacionadas. Esto con el fin de disminuir costos.
Configurar el servicio de Kubernetes para desplegar las aplicaciones de la entrega 1.
Configurar y desplegar las aplicaciones de la entrega 1 en el servicio de Kubernetes.
Construir la documentación técnica del proyecto.

¿Qué debería hacer la semana 5?

Finalizar documento del diseño de la solución para los requerimientos solicitados.
Implementar nuevos microservicios para cumplir con la solución esperada, incluyendo pruebas.
Configurar despliegue de nuevos microservicios.
Configurar el ingress.

¿Qué debería hacer la semana 6?

Finalizar el despliegue de los nuevos microservicios y verificar que los pipelines de evaluación son exitosos.
Completar la documentación, archivos de configuración y demás entregables.
Crear el video y el release.

Recursos:

Usted cuenta con talleres y ejemplos en el contenido del curso. Otros recursos que pueden ser de utilidad para la entrega 2 los puede encontrar en la página Recursos entrega 2

Recomendaciones finales

Revise la rúbrica antes de entregar.
Asegúrese de que todos los pipelines se ejecuten exitosamente.
Valide que la documentación sea clara y completa.
Recuerde que el trabajo en equipo y la planificación son clave para el éxito del proyecto.
Pregunte si tiene duda, no asuma cosas.

Rúbrica

Rúbrica de la entrega 2

Esta rúbrica depende totalmente del archivo `config.yaml` el cuál es el archivo más importante en el repositorio. Este archivo contiene la configuración que se usa en los pipelines para evaluar su entrega y se define la calificación de cada miembro del equipo. Si este archivo no está correctamente configurado, su entrega no puede ser calificada y su nota será de cero.

Entregable	Elemento	Máxima calificación	Criterio de calificación	Calificación por criterio

Presentación	Video.	6	El video se encuentra en la descripción del release, cumple con el tiempo definido y presenta una demo del funcionamiento satisfactorio de todas las aplicaciones haciendo uso de la colección de postman.	
			El video se encuentra en la descripción del release, presenta parcialmente el funcionamiento de las aplicaciones o supera el tiempo permitido.	3
			No se hace la entrega del video o no presenta el funcionamiento de las aplicaciones.	
Documentación	Diseño RF003	10	La documentación se presenta en la página de github pages e incluye el modelo que presenta los nuevos componentes y describe los patrones correctos para solucionar el requerimiento. Indica como serán desplegados los componentes y cumple con las restricciones definidas.	

			La documentación no contiene el diagrama que describe el paso a paso para resolver el requerimiento. O el diagrama no es claro o incorrecto. (resta 2 a la nota del entregable).	
			La documentación no contiene la explicación del patrón que permite resolver el requerimiento. O la explicación no es clara o incorrecta. (resta 2 a la nota del entregable).	
			La documentación no contiene la explicación de los nuevos componentes necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
			La documentación no actualiza el modelo de componentes que permita entender como se relacionan los componentes necesarios para resolver el requerimiento. O estos no son claros o incorrectos.	

			(resta 2 a la nota del entregable).	
			La documentación no actualiza el modelo de despliegue mostrando los componentes y los servicios necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
			No se encuentra la documentación para resolver el primer requerimiento. O esta no es clara.	
	Diseño RF004	10	La documentación se presenta en la página de github pages e incluye el modelo que presenta los nuevos componentes y describe los patrones correctos para solucionar el requerimiento. Indica como serán desplegados los componentes y cumple con las restricciones definidas.	

		La documentación no contiene el diagrama que describe el paso a paso para resolver el requerimiento. O el diagrama no es claro o incorrecto. (resta 2 a la nota del entregable).	
		La documentación no contiene la explicación del patrón que permite resolver el requerimiento. O la explicación no es clara o incorrecta. (resta 2 a la nota del entregable).	
		La documentación no contiene la explicación de los nuevos componentes necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
		La documentación no actualiza el modelo de componentes que permita entender como se relacionan los componentes necesarios para resolver el requerimiento. O estos no son claros o incorrectos.	

			(resta 2 a la nota del entregable).	
			La documentación no actualiza el modelo de despliegue mostrando los componentes y los servicios necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
			No se encuentra la documentación para resolver el primer requerimiento. O esta no es clara.	
	Diseño RF005	10	La documentación se presenta en la página de github pages e incluye el modelo que presenta los nuevos componentes y describe los patrones correctos para solucionar el requerimiento. Indica como serán desplegados los componentes y cumple con las restricciones definidas.	

		La documentación no contiene el diagrama que describe el paso a paso para resolver el requerimiento. O el diagrama no es claro o incorrecto. (resta 2 a la nota del entregable).	
		La documentación no contiene la explicación del patrón que permite resolver el requerimiento. O la explicación no es clara o incorrecta. (resta 2 a la nota del entregable).	
		La documentación no contiene la explicación de los nuevos componentes necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
		La documentación no actualiza el modelo de componentes que permita entender como se relacionan los componentes necesarios para resolver el requerimiento. O estos no son claros o incorrectos.	

		(resta 2 a la nota del entregable).	
		La documentación no actualiza el modelo de despliegue mostrando los componentes y los servicios necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
		No se encuentra la documentación para resolver el primer requerimiento. O esta no es clara, o solo está el modelo sin documentación.	
README	3	El archivo README fue actualizado con la nueva estructura del proyecto y el paso a paso de como desplegar todas sus aplicaciones en la nube.	
		El archivo README no fue actualizado con la nueva estructura del proyecto y el paso a paso de como	

			desplegar todas sus aplicaciones en la nube.	
Implementación	Requerimiento RF003	15	Los componentes trabajan en conjunto para resolver el requerimiento RF003, cumpliendo con el API definido, las restricciones dadas y ejecutándose sobre k8s en el proveedor de nube definido. La nota se calcula con base en el porcentaje de éxito del pipeline ci_evaluador_entrega2.yml. En el caso que el pipeline haya sido modificado la nota será de 0.	
	Consistencia Requerimiento RF003	4	El sistema garantiza la consistencia en caso de indisponibilidad de servicios y retorna el mensaje correcto al usuario final. Se calcula con base en el porcentaje de éxito de los pasos en el pipeline ci_evaluador_entrega2.yml. En el caso que el pipeline haya sido modificado la nota será de 0.	

	Requerimiento RF004	15	Los componentes trabajan en conjunto para resolver el requerimiento RF004, cumpliendo con el API definido, las restricciones dadas y ejecutándose sobre k8s en el proveedor de nube definido. La nota se calcula con base en el porcentaje de éxito del pipeline ci_evaluador_entrega2.yml. En el caso que el pipeline haya sido modificado la nota será de 0.	
	Consistencia Requerimiento RF004	4	El sistema garantiza la consistencia en caso de indisponibilidad de servicios y retorna el mensaje correcto al usuario final. Se calcula con base en el porcentaje de éxito de los pasos en el pipeline ci_evaluador_entrega2.yml. En el caso que el pipeline haya sido modificado la nota será de 0.	

	Requerimiento RF005	15	Los componentes trabajan en conjunto para resolver el requerimiento RF005, cumpliendo con el API definido, las restricciones dadas y ejecutándose sobre k8s en el proveedor de nube definido. La nota se calcula con base en el porcentaje de éxito del pipeline ci_evaluador_entrega2.yml. En el caso que el pipeline haya sido modificado la nota será de 0.	
	Consistencia Requerimiento RF005	4	El sistema garantiza la consistencia en caso de indisponibilidad de servicios y retorna el mensaje correcto al usuario final. Se calcula con base en el porcentaje de éxito de los pasos en el pipeline ci_evaluador_entrega2.yml. En el caso que el pipeline haya sido modificado la nota será de 0.	
Pruebas	Pruebas unitarias de la aplicación	1	El código de gestión de usuarios tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el	

	de gestión de usuarios.		pipeline ci_evaluador_unit.yml.	
			El código de gestión de usuarios no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	
	Pruebas unitarias de la aplicación de gestión de publicaciones.	1	El código de gestión de publicaciones tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el pipeline ci_evaluador_unit.yml.	
			El código de gestión de publicaciones no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	
	Pruebas unitarias de la aplicación	1	El código de gestión de ofertas tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el	

	de gestión de ofertas.		pipeline ci_evaluador_unit.yml.	
			El código de gestión de ofertas no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	
	Pruebas unitarias de la aplicación de gestión de trayectos.	1	El código de gestión de trayectos tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el pipeline ci_evaluador_unit.yml.	
			El código de gestión de trayectos no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	

Requerimientos Funcionales

Como parte de la primera versión del sistema nos enfocaremos en la construcción de una aplicación backend que responde a los siguientes requerimientos funcionales.

Este es un listado de los requerimientos funcionales para una primera versión del proyecto. Sin embargo, es posible que estos requerimientos sean modificados o nuevos sean agregados.

Código	Historia
RF-001	Como usuario deseo crear una cuenta en la aplicación y así poder acceder posteriormente.
RF-002	Como usuario deseo autenticarme en la aplicación con mi nombre de usuario y contraseña, y así crear una sesión para hacer uso de los servicios.
RF-003	Como usuario deseo crear publicaciones para que otros usuarios puedan ofertar por ellas.
RF-004	Como usuario deseo ofertar sobre alguna publicación de otro usuario para poder contratar un servicio.
RF-005	Como usuario deseo que cuando consulte una de mis publicaciones, está contenga de manera ordenada descendente por utilidad, las ofertas que han realizado otros usuarios y así no invertir tiempo innecesario eligiendo la mejor.

RF-001

Código	RF-001

Historia	Como usuario deseo crear una cuenta en la aplicación y así poder acceder posteriormente.
Criterios de aceptación	El usuario introduce la información de su perfíl El sistema válida que el usuario sea único, sino lo es muestra un error. El sistema confirma que el usuario fue creado y ya puede acceder. Si la información del usuario no está completa se retorna un error.

RF-002

Código	RF-002
Historia	Como usuario deseo autenticarme en la aplicación con mi nombre de usuario y contraseña, y así crear una sesión para hacer uso de los servicios.
Criterios de aceptación	El usuario introduce su nombre de usuario y contraseña. El sistema válida las credenciales, si los datos son incorrectos se presenta un error. Si las credenciales son correctas, se genera un token para el usuario.

RF-003

Código	RF-003
Historia	Como usuario deseo crear publicaciones para que otros usuarios puedan ofertar por ellas.
Criterios de aceptación	El usuario brinda los datos de la publicación y el trayecto que hará. Se valida si el trayecto ya existe o no, sino existe se crea con los valores indicados y se usa ese para la creación de la publicación, si ya existe se descartan los datos brindados y se usa el que está almacenado. La publicación queda asociada al usuario de la sesión. La plataforma solo permite crear publicaciones cuando la fecha de inicio de viaje es en el futuro. La plataforma solo permite crear publicaciones cuando la fecha de expiración de la publicación es posterior a la fecha actual y anterior o igual a la fecha de inicio de viaje. Si el usuario ya tiene otra publicación para el mismo trayecto, se rechaza la creación. Solo un usuario autenticado puede realizar esta operación. En cualquier caso de error la información en las bases de datos debe ser consistente al finalizar.

RF-004

Código	RF-004
Historia	Como usuario deseo ofertar sobre alguna publicación de otro usuario para poder contratar un servicio.

Criterios de aceptación	El usuario brinda la información de la oferta que desea hacer y el identificador de la publicación a la que se realiza. Se valida que la publicación existe, solo se puede crear la oferta en una publicación existente. Solo es posible crear la oferta si la publicación no ha expirado. La oferta queda asociada al usuario de la sesión. El usuario no debe poder ofertar en sus publicaciones. Se calcula la utilidad (score) de la oferta. Solo un usuario autenticado puede realizar esta operación. En cualquier caso de error la información en las bases de datos debe ser consistente al finalizar.

RF-005

Código	RF-005
Historia	Como usuario deseo que cuando consulte una de mis publicaciones, está contenga de manera ordenada descendente por utilidad, las ofertas que han realizado otros usuarios y así no invertir tiempo innecesario eligiendo la mejor.

Criterios de aceptación	El usuario brinda el identificador de la publicación que desea consultar y retorna la información de la publicación, sus ofertas, y el trayecto asociado. Las ofertas se encuentran ordenadas de forma descendente por la utilidad que brindan. El valor de la utilidad se encuentra en la respuesta junto con el detalle de cada oferta. Si el usuario no es el dueño de la publicación se debe presentar un error. Solo el propietario debe ver la información de sus propias publicaciones. Si no es posible obtener la oferta temporalmente (indisponibilidad) se debe presentar el valor "null".

Supuestos

En el alcance de este proyecto solo se tendrán en cuenta los trayectos sin escalas.

TABLE OF CONTENTS

Creación de Publicaciones

En este documento se especifica el endpoint que nuestras aplicaciones cliente (imagine que tenemos aplicaciones web y móvil) utilizarán para cumplir con el requerimiento RF-003 de creación de publicaciones:

API

Descripción

La siguiente es la definición del endpoint que permite crear una Publicación cumpliendo con los criterios de aceptación definidos para el requerimiento RF-003.

Método	POST
Ruta	/rf003/posts
Parámetros	
Encabezados	`Authorization: Bearer token`

Cuerpo	`{`
	`"flightId": identificador del vuelo,`
	`"expireAt": fecha y hora máxima en la que se recibirán ofertas sobre la publicación en formato ISO,`
	`"plannedStartDate": fecha y hora planeada de salida del origen en formato ISO,`
	`"plannedEndDate": fecha y hora planeada de llegada en formato ISO,`
	`"origin": {`
	`"airportCode": código del aeropuerto de origen,`
	`"country": nombre del país de origen`
	`},`
	`"destiny": {`
	`"airportCode": código del aeropuerto de origen,`
	`"country": nombre del país de origen`
	`},`
	`"bagCost": costo de envío de maleta en dólares`
	`}`

Respuestas

Código	Descripción	Cuerpo

400	En el caso que alguno de los campos no esté presente en la solicitud, o no tenga el formato esperado.	
401	El token no es válido o está vencido.	
403	El token no está en la solicitud.	

412	En el caso que la fecha de inicio y fin del trayecto no sean válidas; fechas en el pasado o no consecutivas.	{
		"msg": "Las fechas del trayecto no son válidas"
		}

412	En el caso que la fecha de expiración no sea en el futuro o no sea válida.	{
		"msg": "La fecha expiración no es válida"
		}

412	Si el usuario ya tiene otra publicación para el mismo trayecto.	{
		"msg": "El usuario ya tiene una publicación para la misma fecha"
		}

201	Si la creación de la publicación es exitosa.	{
		"data": {
		"id": id de la publicación,
		"userId": id del usuario que crea la publicación,
		"createdAt": fecha y hora de creación de la publicación en formato ISO,
		"expireAt": fecha y hora del último día en que se reciben ofertas sobre la publicación,
		"route": {
		"id": id del trayecto,
		"createdAt": fecha y hora de creación del trayecto en formato ISO,
		}
		},
		"msg": Resumen de la operación *.
		}

503	Si durante el proceso alguna integración falla y el proceso no	{
		"msg": "El servicio está temporalmente fuera de servicio."
		}

	puede terminar satisfactoriamente.

> El resumen de la operación puede contener aún más detalle de la transacción, es de libre uso por parte del equipo pero no puede ser vacío.

Creación de Ofertas

En este documento se especifica el endpoint que nuestras aplicaciones cliente (imagine que tenemos aplicaciones web y móvil) utilizarán para cumplir con el requerimiento RF-004 de creación de ofertas:

API

Descripción

La siguiente es la definición del endpoint que permite crear una Oferta cumpliendo con los criterios de aceptación definidos para el requerimiento RF-004.

Método	POST
Ruta	/rf004/posts/{id}/offers
Parámetros	id: identificador de la publicación a la que se quiere asociar la oferta.
Encabezados	`Authorization: Bearer token`

Cuerpo	{
	"description": descripción del paquete a llevar,
	"size": LARGE ó MEDIUM ó SMALL,
	"fragile" : booleano que indica si es un paquete delicado o no,
	"offer": valor en dólares de la oferta para llevar el paquete
	}

Respuestas

Código	Descripción	Cuerpo
400	En el caso que alguno de los campos no esté presente en la solicitud.	
401	El token no es válido o está vencido.	
403	El token no está en la solicitud.	
404	La publicación a la que se quiere asociar la oferta no existe.	

412	La publicación es del mismo usuario y no se puede ofertar por ella.	
412	La publicación ya está expirada y no se reciben más ofertas por ella.	
201	Si la creación de la oferta es exitosa. La utilidad de la oferta queda almacenada en la base de datos del servicio de utilidad.	{ "data": { "id": id de la oferta, "userId": id del usuario dueño de la oferta, "createdAt": fecha de creación de la oferta, "postId": id de la publicación }, "msg": Resumen de la operación *. }
503	Si durante el proceso alguna integración falla y el proceso no puede terminar satisfactoriamente.	{ "msg": "El servicio está temporalmente fuera de servicio." }

> El resumen de la operación puede contener aún más detalle de la transacción, es de libre uso por parte del equipo pero no puede ser vacío.

Consultar Publicaciones

En este documento se especifica el endpoint que nuestras aplicaciones cliente (imagine que tenemos aplicaciones web y móvil) utilizarán para cumplir con el requerimiento RF-005 de consulta de publicaciones:

API

Descripción

La siguiente es la definición del endpoint que permite consultar una Publicación cumpliendo con los criterios de aceptación definidos para el requerimiento RF-005.

Método	GET
Ruta	/rf005/posts/{id}
Parámetros	id: identificador de la publicación a consultar
Encabezados	`Authorization: Bearer token`
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
401	El token no es válido o está vencido.	
404	La publicación no existe.	
403	El usuario no tiene permiso para ver el contenido de esta publicación.	

200	Objeto con todos los datos de una publicación.	{
		"data": {
		"id": identificador de la publicación,
		"expireAt": fecha y hora máxima en que se reciben ofertas en formato ISO,
		"route": {
		"id": identificador del trayecto,
		"flightId": identificador del vuelo,
		"origin": {
		"airportCode": código del aeropuerto de origen,
		"country": nombre del país de origen
		},
		"destiny": {
		"airportCode": código del aeropuerto de destino,
		"country": nombre del país de destino
		},
		"bagCost": costo de envío de maleta
		},
		"plannedStartDate": fecha y hora en que se planea el inicio del viaje en formato ISO,
		"plannedEndDate": fecha y hora en que se planea la finalización del viaje en formato ISO,
		"createdAt": fecha y hora de creación de la publicación en formato ISO,
		"offers": [
		{
		"id": identificador de la oferta,
		"userId": identificador del usuario que hizo la oferta,
		"description": descripción del paquete a llevar,
		"size": LARGE ó MEDIUM ó SMALL,
		"fragile": booleano que indica si es un paquete delicado o no,
		"offer": valor en dólares de la oferta para llevar el paquete,
		"score": utilidad que deja llevar este paquete en la maleta,
		"createdAt": fecha y hora de creación de la oferta en formato ISO
		}
		]
		}
		}

503	Si durante el proceso alguna integración falla y el proceso no puede terminar satisfactoriamente.	<pre>{
		"msg": "El servicio está temporalmente fuera de servicio."
		}</pre>

Restricciones

Las siguientes restricciones se tienen que cumplir en la implementación. El incumplimiento de estas restricciones implica la no calificación de las entregas.

> Este documento puede estar siendo actualizado durante el desarrollo del curso, cualquier cambio será comunicado previamente a todos los estudiantes por los canales definidos.

Restricciones generales

Este proyecto se trabajará única y exclusivamente en los repositorios brindados por la dirección del curso. Si no ha sido agregado al repositorio, favor comuníquese con el tutor por medio de slack.
La estrategia para el proyecto es Monorepo. Haga uso de un solo repositorio de GitHub para almacenar el código fuente de todos los servicios. Cada componente debe estar en una carpeta aparte y no debe haber acoplamiento entre ellas.
El video de la entrega debe estar embebido dentro de la descripción del release, no puede estar publicado en otras plataformas ni se puede entregar por medio

de enlaces. Si el video no se encuentra en la descripción del release, la nota de este será de cero.

Restricciones de la segunda entrega

Usar PostgreSQL como base de datos. Otras bases de datos no deben ser utilizadas a menos que se indique de manera explicita en el enunciado.
    El puerto de la base de datos DEBE ser `5432`.
Las fechas deben manejarse en formato ISO `yyyy-mm-ddTHH:MM:SS`. La zona horaria será UTC 0.
Todas las aplicaciones construidas en la entrega 1 se deben utilizar.
Las aplicaciones construidas en la entrega 1 NO se deben modificar. Solo se permitirán cambios que sean aprobados por el equipo de tutores, siempre y cuando entre sus modificaciones no se altere la entidad y campos con los que trabajan, y tampoco su arquitectura (no se pueden agregar nuevas integraciones a otros microservicios). En el caso que se evidencie un cambio que no fue aprobado, la nota de la entrega será de cero. Para recibir la aprobación de un cambio en las aplicaciones de la entrega 1, usted DEBE:
    Crear un PR a la rama `main` y poner a los tutores como revisores del PR, indicando la razón del cambio. Puede mezclar el cambio sin recibir la aprobación, el tutor puede revisar y aprobar el cambio posteriormente. Tenga presente que si el tutor no lo aprueba, se aplicará la penalidad descrita en este documento.
    Crear un archivo `CHANGELOG.md` en la raiz del repositorio y describir el cambio con los enlaces a los PRs correspondientes. Siga el formato definido en Keep a Changelog.
La entrega debe ejecutarse en el proveedor de nube seleccionado y será calificada solo sobre este ambiente, la ejecución exitosa en otros ambientes no es válido para su calificación.
El lenguaje de programación es de libre elección del equipo, pero en cualquier lenguaje se debe cumplir con el 70% de cobertura del código. Antes de iniciar el desarrollo debe validar con el equipo de tutores el lenguaje y la librería para verificar la cobertura del código.
Tanto en K8s como en Terraform no se deben usar namespaces, siempre haga uso de los namespaces por defecto.
Para todo despliegue en AWS usaremos la región `us-east-1`. Despliegues en otras regiones no serán válidos.

Recursos para la entrega 2

Código

> Haga uso de los recursos de la Entrega 1 para la construcción de aplicaciones. Los talleres vistos durante estas semanas le enseñan cómo desplegar en la nube usando Terraform y K8s.

> Le recomendamos crear un archivo de scripts para desplegar siempre su infraestructura de manera replicable. Por ejemplo, usar `make`.

Pruebas

Para probar sus aplicaciones localmente y validar que responden correctamente a la especificación de API REST, puede usar las siguientes colecciones. Descargue cada archivo JSON e impórtelo en su cliente de Postman. Estas mismas pruebas son usadas para la calificación de su entrega.

> Pruebas para RF003:
> https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/proyecto-monitor/main/entrega2/evaluate_rf003.json
> Pruebas para RF004:
> https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/proyecto-monitor/main/entrega2/evaluate_rf004.json
> Pruebas para RF005:
> https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/proyecto-monitor/main/entrega2/evaluate_rf005.json

Las siguientes colecciones le permiten validar si su sistema garantiza la consistencia en el caso de la indisponibilidad de una aplicación. Para usarlas localmente tiene que bloquear la comunicación entre aplicaciones.

> Pruebas para consistencia de RF003:
> https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/proyecto-monitor/main/entrega2/evaluate_rf003_consistency.json

Pruebas para consistencia de RF004:
https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/proyecto-monitor/main/entrega2/evaluate_rf004_consistency.json
Pruebas para consistencia de RF005:
https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/proyecto-monitor/main/entrega2/evaluate_rf005_consistency.json

Para importar la colección use el botón Import en la sección superior izquierda.

Una vez importada la colección, actualice la variable de ambiente `BASE_PATH` por el host de su ingress ya sea usando minikube o su despliegue en la nube.

Finalmente, ejecute la colección haciendo clic derecho en el nombre de la colección y seleccionando `Run`. Si la ruta de la aplicación está correctamente configurada Postman ejecutará múltiples solicitudes de API y también ejecutará algunos assertions que hemos preparado para asegurarnos de que la aplicación esté funcionando como se espera.

Pipelines

En su repositorio de equipo encontrará una serie de pipelines para verificar que su entrega cumple con parte de los entregables calificables. Estos pipelines se ejecutan como un workflow de Github Actions de forma manual o automática dependiendo del workflow.

El workflow que ejecuta las pruebas automáticas se ejecuta cada que usted mezcle en la rama `main` o cuando haga un Pull Request a la rama `main`. Ese workflow no debería ser modificado para esta entrega y su ejecución solo confirma que todo lo hecho en la entrega 1 sigue funcionando como se espera.

Encontrará en el menú izquierdo una lista de todos los flujos de trabajo (workflows) disponibles para ejecución. En este caso, verá `Evaluador Implementación Entrega 2` que corresponde al pipeline de pruebas de sus aplicaciones sobre su despliegue en la nube. Haga clic en el workflow y verá un botón `Run workflow` en la sección superior derecha. Seleccione el ambiente de ejecución y la rama en la que desea ejecutarlo, por último oprima el botón `Run workflow`.

Tecnologías del proyecto

El equipo de desarrollo podrá seleccionar los lenguajes con los que desarrollarán las aplicaciones, sin embargo, las tecnologías de ejecución y despliegue, así como las tecnologías de bases de datos, no serán opcionales y solo podrán usarse las ya definidas.

En esta entrega continuaremos usando las tecnologías descritas en la Entrega 1 y se agregan las siguientes:

Tecnologías requeridas desde la segunda entrega

AWS Academy. Plataforma educativa para la ejecución de ambientes de AWS.
AWS CLI. Cliente de AWS para interactuar con el API de AWS.
Terraform. Nuestra herramienta para infrastructura como código.
Tfenv. Herramienta de gestión de diferentes versiones de terraform.
EKS
ECR
SecretsManager
RDS

> Nota: Les recomendamos estar vigilando los créditos disponibles en su cuenta con el fin de evitar un consumo total antes de realizar la entrega del proyecto.

Entrega 3

Comprende los temas vistos hasta la semana 6 y su fecha límite es el sábado de la semana 8 11:59 pm GMT - 5.

Table of contents

Enunciado de la entrega 3

TABLE OF CONTENTS

Objetivos

> Evaluar el conocimiento de los estudiantes para resolver problemas de negocio haciendo uso de arquitecturas serverless y dirigida por eventos, siguiendo buenas prácticas de desarrollo y calidad de código.
> Implementar servicios de envío de mensajes, patrones para la entrega de mensajes y servicios Serverless de un proveedor de nube para resolver problemas de coreografía.

El objetivo de su equipo en esta entrega es documentar, diseñar, implementar y desplegar los componentes necesarios para cumplir con una serie de requerimientos. Cabe resaltar que debe hacer uso del conjunto de servicios ya implementados y los patrones de diseño vistos durante estas semanas.

Prerrequisitos y restricciones

La entrega debe cumplir con las restricciones descritas en el documento de restricciones y utilizar las tecnologías listadas en el documento de tecnologías.

Para esta entrega NO es necesario que todos los componentes desarrollados en entregas anteriores estén desplegados, le recomendamos sólo dejar en ejecución los componentes qué considere necesarios.

Alcance

El equipo fundador se ha reunido con el equipo de ingeniería para ultimar detalles sobre los siguientes pasos en la construcción de nuestra plataforma. En esta reunión se ha llegado a la conclusión de trabajar en algunos requerimientos de suma importancia y de baja complejidad adecuados para este ciclo del desarrollo. Los requerimientos son:

- RF-006
- RF-007

> Favor lea los criterios de aceptación de cada uno de los requerimientos para tener claridad de como se calificará la entrega.

Para lograr estos requerimientos el equipo técnico ha decidido contratar los servicios de la empresa TrueNative, una plataforma que ofrece servicios de verificación de datos y gestión de pagos. Le recomendamos revisar inicialmente la documentación de TrueNative para entender a detalle sus características y funcionamiento.

Para estos nuevos requerimientos debe crear una nueva entidad: Tarjeta de Crédito (CreditCard). Esta debe ser desplegada como un componente único y totalmente desacoplado de otros, y se debe implementar su API completo así como su modelo de datos. El diseño del API y el modelo de datos se encuentran definidos en los siguientes links:

- Diseño de la entidad
- Documentación de la API

Todos los componentes deben ser desplegados sobre un proveedor de nube haciendo uso de los servicios de bases de datos, serverless y orquestación vistos durante las últimas semanas.

Al finalizar cualquier de los procesos de verificación, el usuario debe ser notificado por medio de correo electrónico.

Los entregables de esta entrega son:

Documentación

Haciendo uso de la información de los requerimientos, descripción de los servicios ya implementados en el sistema (entrega 1) y de la plataforma TrueNative, el equipo de trabajo debe construir un documento de arquitectura donde se describen los componentes y decisiones de diseño que solucionan los dos requerimientos de esta entrega. Esta documentación debe estar almacenada y debe ser navegable en la carpeta `/docs`.

El diseño debe contemplar las siguientes restricciones de diseño.

La solución debe contemplar que el usuario final solo conocerá UN punto de acceso al sistema (un solo host) y por lo tanto hará UNA sola petición HTTP por cada requerimiento, y no una secuencia. El sistema recibirá cada solicitud y se encargará de que todos los componentes trabajen juntos para completar cada objetivo.

La solución debe contemplar una alta demanda de usuarios que se incrementará con el tiempo.

La solución debe contemplar que solo usuarios autenticados y autorizados pueden hacer uso de estos nuevos servicios.

La solución debe contemplar todo lo que se encuentra implementado. No se deben duplicar funcionalidades.

La solución debe contemplar que trabajamos con microservicios y no debemos tener acoplamientos que nos impacten la escalabilidad, disponibilidad, facilidad de desecho y mantenibilidad de nuestras aplicaciones.

La solución debe contemplar que parte del proceso debe realizarse de forma asíncrona y por lo tanto debe hacer uso de patrones y componentes de una arquitectura dirigida por eventos.

La solución debe hacer uso de TrueNative y tener en cuenta sus limitaciones actuales.

Las funcionalidades actuales deben continuar funcionando sin ser alteradas, a excepción del microservicio de usuarios que ahora no podrá generar ni verificar tokens si el usuario no está verificado. - Los enlaces a documentación externa están totalmente prohibidos, toda la documentación y modelos deben estar contenidos en la misma carpeta, incluyendo las imágenes que deben ser creados usando PlantUML. - La documentación debe contener las siguientes elementos:

Un modelo de componentes de la vista funcional en el que se describen los componentes necesarios (tanto nuevos como anteriormente implementados) para cumplir con los requerimientos. Cada componente NUEVO debe estar documentado usando la siguiente tabla:

Componente	Nombre del componente
Código/Id del componente	Debe aparecer en el diagrama
Tipo	Tipo de componente: base de datos, servicio, almacenamiento, etc
Responsabilidad	Responsabilidad del componente dentro del sistema/aplicación
Consideraciones de diseño	Tradeoffs, impacto en el sistema actual, puntos de dolor, etc.
Integraciones	Con cuáles componentes se comunica, protocolos, verbos, tipo de comunicación

Un modelo de despliegue de la vista de despliegue en el que se describen como los componentes funcionales serán desplegados haciendo uso de los servicios de la nube.
Un módelo de secuencia por cada requerimiento donde se describa el proceso que soluciona el requerimiento. Este debe tener en cuenta flujos de excepción y manejo de errores

Una página donde se describen los patrones con los que se resuelve cada requerimiento. Las tablas deben tener este formato:

Requerimiento	Identificador del requerimiento.
Patrón utilizado	Nombre del patrón utilizado.
Justificación	Justificación de por que este patrón puede ser utilizado y para qué se debe usar.
Atributos de calidad favorecidos	Lista de atributos de calidad que se favorecen al utilizar este patrón en este contexto.
Atributos de calidad desfavorecidos	Lista de atributos de calidad que se desfavorecen al utilizar este patrón en este contexto.
Componentes involucrados	Lista de los componentes involucrados del modelo para resolver el requerimiento (nombre, código).

La nueva documentación debe estar publicada en su página web y debe ser fácilmente navegable por medio de la tabla de contenido. Por ejemplo:

```
# Documentación técnica

...

 * [Vista de componentes](ruta a la página con la documentación).
```

```
  * [Vista de despliegue](ruta a la página con la documentación).

  * [Vista de desarrollo](ruta a la página con la documentación).

    * [Descripción de nuevos componentes](ruta a la página con la documentación).

  * Requerimientos

    ...

    * Requerimiento RF-006

      * [Documentación patrón solución requerimiento 6](ruta a la página con la documentación).

      * [Explicación del proceso paso a paso de la solución requerimiento 6](ruta a la página con la documentación).

    * Requerimiento RF-007

      * [Documentación patrón solución requerimiento 7](ruta a la página con la documentación).

      * [Explicación del proceso paso a paso de la solución requerimiento 4](ruta a la página con la documentación).
```

> Actualicen el archivo `README.md` en la raíz del proyecto que indique la estructura de las carpetas y cómo hacer el despliegue.
> Actualicen el archivo `config.yaml` con la siguiente estructura. (Este archivo es necesario para las pruebas por parte del equipo calificador y su entrega es obligatoria).

```
team:

    name: <nombre del equipo>

    members:

        - user: <usuario uniandes>

          percentage: <0 a 100>

        - user: <usuario uniandes>

          percentage: <0 a 100>
```

```
        - user: <usuario uniandes>
          percentage: <0 a 100>
        - user: <usuario uniandes>
          percentage: <0 a 100>
  board: <link al dashboard de tareas del equipo>
  docs: <link al github pages del equipo>
  url: <url del host de la aplicación, ejemplo https://host>
  cluster_name: <Nombre del cluster de K8s>
  users_app:
    ...
  posts_app:
    ...
  offers_app:
    ...
  routes_app:
    ...
  scores_app:
    ...
  <nuevo componente>:
    folder: <nombre de la carpeta del nuevo componente>
    type: <Tipo de aplicación: contenedora o función>
    authors:
      - <usuario uniandes>
  ... # otros componentes (Todos los componentes deben estar listados acá).
```

El archivo `config.yaml` es el archivo más importante en el repositorio. Este archivo contiene la configuración que se usa en los pipelines para evaluar su entrega y se define la calificación de cada miembro del equipo. Tenga en cuenta que, si este archivo no está correctamente configurado, no será posible calificar la entrega.

Formato de entrega

- Archivos markdown en el repositorio del equipo en el release `ProyectoTerceraEntrega`.
- Página web desplegada en github pages y creada a partir de los archivos en la carpeta `/docs`.

Este entregable se califica sobre un release que debe ser creado en el repositorio de su equipo y que debe llevar el nombre `ProyectoTerceraEntrega`. El release debe ser creado antes de la hora límite de entrega del proyecto. Si el release es creado posterior a la hora límite, se aplicarán las sanciones descritas en la sección de calificación del curso. La calificación se realizará con base en el contenido del release marcado como `ProyectoTerceraEntrega`, por lo que cualquier cambio fuera de este no será considerado.

Mecanismo de evaluación

Este entregable se evaluará de dos formas:

- Manualmente, los tutores del curso revisarán a detalle su página web para validar que todas las unidades y los modelos son correctos. Tenga presente que la página web de su proyecto debe ser privada.

Implementación

Siguiendo el diseño documentado, como equipo deben realizar la implementación funcional de los componentes documentados.

- Código funcional de todas las aplicaciones del sistema siguiendo la documentación creada por el equipo.
  - Se seguirá usando el mismo repositorio de código de Github de la primera entrega. Todo el código debe quedar sobre la rama master o main, no se calificará en otras ramas.

- Cada componente puede ser construido usando contenedores o funciones como servicio en el lenguaje de su preferencia; Python, JavasCript, Java, etc. Valide con su tutor la tecnología que desea usar en el caso que no use la recomendada por el curso (Python).
- Puede hacer uso o no de bases de datos no relacionales para la implementación de la nueva entidad (CreditCard).
- Continue con la misma estructura de carpetas que trabajó en la entrega anterior, en el que cada aplicación debe estar almacenada en su propia carpeta.
- Cada componente está formado por el código fuente, sus pruebas unitarias, un archivo `README.md` con la descripción del componente y el archivo Dockerfile si lo requiere.
- El correo electrónico de verificación debe llegar a la dirección de correo del usuario que realiza la solicitud. Su solución no debe funcionar para un segmento de correos específico. Los usuarios de pruebas pueden usar correos temporales y efímeros.
- El formato de los correos electrónicos es decisión del equipo, pero el contenido debe ser el acordado en los criterios de aceptación de cada requerimiento.
- Puede usar la librería o servicio que desee para el envío de correos electrónicos, no hay restricción sobre esta.

Las aplicaciones deben debe ser desplegadas en contenedores haciendo uso de Docker y orquestadas por medio de Kubernetes sobre un proveedor de nube, o desplegadas como funciones sobre AWS Lambda.

- Todos los componentes deben ser desplegados sobre los servicios del proveedor de nube seleccionado, NO sobre un ambiente local. Ningún componente será calificado sobre un despliegue local.
- En la carpeta `k8s` se deben encontrar los archivos necesarios para el despliegue de los componentes de Kubernetes. Documente la estructura de sus archivos en la vista de desarrollo del proyecto.
- En la carpeta `k8s` se debe encontrar el archivo `k8s-true-native-deployment.yaml` que despliega la aplicación TrueNative.

Por medio de la URL de su sistema se debe poder acceder a los endpoints que procesan los requerimientos del enunciado y al API de TrueNative. Se validarán que todas las rutas sean posibles de acceder, sin excepción. Si algunas de estas rutas no puede ser accedido por la URL, se tomará como que la entrega está incompleta y se penitenciará la entrega. La estructura de endpoints debe ser la siguiente:

`https://<host>/users/*`

`https://<host>/credit-cards/*`

`https://<host>/native/* (true native)`

Formato de entrega

Link del release `ProyectoTerceraEntrega` creado antes de la fecha y hora máxima de entrega.

Este entregable se califica sobre un release que debe ser creado en el repositorio de su equipo y que debe llevar el nombre `ProyectoTerceraEntrega`. El release debe ser creado antes de la hora límite de entrega del proyecto. Si el release es creado posterior a la hora límite, se aplicarán las sanciones descritas en la sección de calificación del curso. La calificación se realizará con base en el contenido del release marcado como `ProyectoTerceraEntrega`, por lo que cualquier cambio fuera de este no será considerado.

Mecanismo de evaluación

Este entregable se evaluará de dos formas:

- Por medio de pipelines. En su repositorio encontrará los pipelines `ci_evaluador_entrega3.yml` y `ci_evaluador_entrega3_email.yml`, los cuales deben ejecutarse de manera exitosa sobre el último commit de su release. Si algunos de los archivos es modificado, se penitenciará como indica la rúbrica. El pipeline realiza las siguientes tareas:
  - Ejecuta pruebas de API sobre cada endpoint para validar que la aplicación cumple con el requerimiento de validación de identidad.
  - Ejecuta pruebas de API sobre cada endpoint para validar que la aplicación cumple con el requerimiento de creación y verificación de tarjetas de crédito.
  - Prueba el envío de correos electrónicos creando un usuario con un correo electrónico que se agrega como entrada del pipeline.
- Manualmente se verificará que la solución entregada sea acorde a las decisiones de diseño planteadas en el enunciado: Componentes, despliegue, estructura. Se evaluará la cohesión y aislamiento; que una aplicación no gestione o tenga acceso a información que debe ser gestionada por otras aplicaciones.

Pruebas

Las pruebas unitarias de las aplicaciones de la primera entrega deben seguir funcionando en la ejecución del pipeline y cubrir el 70% de código.

> El código debe contar con pruebas unitarias que cubran al menos 70% del código de cada aplicación. La ejecución de las pruebas se validará únicamente en pipelines. Si las pruebas no son ejecutadas y la cobertura no es verificada por el pipeline la nota será de cero. NO se admiten ejecuciones locales.
> Se debe modificar el archivo de pipelines con el nombre `ci_evaluador_unit.yml` para que contenga todos los jobs de pruebas unitarias. Cada job debe estar diseñado para probar el cubrimiento de una aplicación, y este no debe ser menor al 70% (En el código fuente del repositorio de su equipo encontrará un ejemplo de un proyecto en Python y como está configurado un pipeline de github).
> Las pruebas unitarias serán opcionales para todas las nuevas aplicaciones agregadas al repositorio. Su implementación no se tomará en cuenta para la calificación de la entrega.

Formato de entrega

> Link del release `ProyectoTerceraEntrega` creado antes de la fecha y hora máxima de entrega.

Entre entregable se califica sobre un release que debe ser creado en el repositorio de su equipo y que debe llevar el nombre `ProyectoTerceraEntrega`. El release debe ser creado antes de la hora límite de entrega del proyecto. Si el release es creado posterior a la hora límite, se aplicarán las sanciones descritas en la sección de calificación del curso. La calificación se realizará con base en el contenido del release marcado como `ProyectoTerceraEntrega`, por lo que cualquier cambio fuera de este no será considerado.

Mecanismo de evaluación

Este entregable se evaluará únicamente por medio de pipelines del repositorio. El archivo `ci_evaluador_unit.yml` debe contener mínimo 4 jobs (uno por aplicación; posts_app, users_app, offers_app, routes_app) en los que se verifica que el cubrimiento de pruebas es exitoso.

Si el equipo utiliza otro lenguaje diferente a Python, u otro framework de pruebas distinto a pytest, deben agregar una sección en su README describiendo las

herramientas que usaron, como funcionan, donde se encuentra la documentación oficial de las tecnologías, y describir ese funcionamiento en su video de sustentación.

Presentación

Grabar un video donde presente su equipo, los detalles más relevantes de su proyecto, y una demo de como funciona el sistema.

> En el video se presenta al equipo que trabajó en la entrega, una descripción del repositorio y un demo de como funciona la aplicación haciendo uso de un cliente como Postman. La duración del video debe estar entre cinco y diez minutos. En caso de exceder el tiempo máximo, se aplicará una penalización en la nota de este entregable.

Formato de entrega

El video debe ser incluido en la descripción de su release `ProyectoTerceraEntrega` en el momento de la creación. El video no puede ser incluido posterior a la creación del release y NO será recibido por ningún otro medio o mecanismo.

Mecanismo de evaluación

El video se evaluará manualmente verificando que describe la estructura del sistema y presenta una demo del funcionamiento de la aplicación. Si el video no cumple con la duración esperada o su calidad no permite evaluar que su contenido es el esperado, el entregable será penitenciado como indica la rúbrica.

Después de la entrega, uno o dos grupos podrán ser seleccionados aleatoriamente por los tutores para realizar una sustentación síncrona en la que deberán exponer su solución y cada uno de los entregables descritos; la nota será calculada con base en esa presentación. Esa sustentación sucederá en los siguientes tres días hábiles a la fecha de entrega.

Calificación

La calificación se hará tomando el éxito de inspecciones manuales y el éxito de los pipelines ejecutados. Tenga presente que el tutor puede, si lo considera necesario,

ejecutar las pruebas en otros ambientes y tomar el resultado de esa ejecución como su nota. Será debidamente notificado en el caso que suceda.

La calificación se realizará tomando como base esta rúbrica.

> Para calificar su entrega usted debe crear el release y posteriormente ejecutar ambos pipelines para validar el funcionamiento. El tutor se encargará de reejecutar pipelines y verificar el envío de correos por lo que es importante que la infraestructura de su solución esté desplegada y disponible. Solo haciendo uso de los pipelines se calificará su entrega.
> Para los equipos que terminaron su entrega antes de la hora/día definidos, se puede adelantar la calificación con el fin de evitar costos adicionales en sus cuentas de AWS. Para ello pueden enviar un correo a ca.forero10@uniandes.edu.co para asignar a un tutor que revise su entrega o comunicándose con un tutor por medio de slack.
> Para los equipos que no logren terminar su entrega antes de la hora y fecha máxima de entrega. Deben dejar la infraestructura encendida hasta el momento de la calificación. Es posible que el tutor se comunique con ustedes para indicarles que debe encender o apagar la infraestructura. Este pendiente y disponible para esta tarea.

> Después de la entrega, uno o dos grupos podrán ser seleccionados aleatoriamente por los tutores para realizar una sustentación síncrona en la que deberán exponer su solución y cada uno de los entregables descritos; la nota será calculada con base en esa presentación. Esa sustentación sucederá en los siguientes tres días hábiles a la fecha de entrega.

Estrategia recomendada

En esta sección se indica cuál es la estrategia recomendada que usted con su equipo deberían de seguir para cumplir con el entregable. Esta solo es una recomendación, es decisión de su equipo las tareas a desarrollar semana a semana.

División de trabajo

Recomendamos que cada miembro del equipo trabaje en una parte del proceso de algún requerimiento. Una persona podría encargarse del primer requerimiento, otros dos del segundo y el último del manejo de notificaciones. Tenga presente que aunque el desarrollo es pequeño, se debe tener en cuenta no solo el código, sino la configuración de otros servicios para su completitud.

¿Qué debería hacer la semana 7?

- Revisar del enunciado de la entrega 3.
- Revisar la documentación de TrueNative, desplegarlo y probarlo sobre su cluster.
- Planear las actividades y la división de responsabilidades. Haga uso del tablero Kanban de su repositorio para planear y gestionar su trabajo.
- Configurar sus nuevas ramas sobre el mismo repositorio.
- Definir como harán uso de las cuentas de AWS para la entrega.
- Realizar el diseño de la solución.
- Iniciar la construcción de los nuevos componentes.

¿Qué debería hacer la semana 8?

- Finalizar la implementación y el despliegue de los nuevos componentes.
- Completar la documentación, archivos de configuración y demás entregables.
- Hacer el video.

Recursos:

Usted cuenta con talleres y ejemplos en el contenido del curso. Otros recursos que pueden ser de utilidad para la entrega 3 los puede encontrar en la página Recursos entrega 3

Recomendaciones finales

Revise la rúbrica antes de entregar.
Asegúrese de que todos los pipelines se ejecuten exitosamente.
Valide que la documentación sea clara y completa.
Recuerde que el trabajo en equipo y la planificación son clave para el éxito del proyecto.
Pregunte si tiene duda, no asuma cosas.

Rúbrica

Rúbrica de la entrega 3

Esta rúbrica depende totalmente del archivo `config.yaml` el cuál es el archivo más importante en el repositorio. Este archivo contiene la configuración que se usa en los pipelines para evaluar su entrega y se define la calificación de cada miembro del equipo. Si este archivo no está correctamente configurado, su entrega no puede ser calificada y su nota será de cero.

Entregable	Elemento	Máxima calificación	Criterio de calificación	Calificación por criterio
Presentación	Video.	6	El video se encuentra en la descripción del release, cumple con el tiempo definido y presenta una demo del funcionamiento satisfactorio de todos los requerimientos haciendo uso de la colección de postman y los pipelines.	

			El video se encuentra en la descripción del release, presenta parcialmente el funcionamiento de las aplicaciones o supera el tiempo permitido.	3
			No se hace la entrega del video o no presenta el funcionamiento de las aplicaciones.	
Documentación	Diseño RF006	10	La documentación se presenta en la página de github pages e incluye el modelo que presenta los nuevos componentes y describe los patrones correctos para solucionar el requerimiento. Indica como serán desplegados los componentes y cumple con las restricciones definidas.	
			La documentación no contiene el diagrama que describe el paso a paso para resolver el requerimiento. O el diagrama no es claro o incorrecto. (resta 2 a la nota del entregable).	

			La documentación no contiene la explicación del patrón que permite resolver el requerimiento. O la explicación no es clara o incorrecta. (resta 2 a la nota del entregable).	
			La documentación no contiene la explicación de los nuevos componentes necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
			La documentación no actualiza el modelo de componentes que permita entender como se relacionan los componentes necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
			La documentación no actualiza el modelo de despliegue mostrando los componentes y los servicios necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	

			No se encuentra la documentación para resolver el requerimiento 6. O esta no es clara, o solo está el modelo sin documentación.	
	Diseño RF007	10	La documentación se presenta en la página de github pages e incluye el modelo que presenta los nuevos componentes y describe los patrones correctos para solucionar el requerimiento. Indica como serán desplegados los componentes y cumple con las restricciones definidas.	
			La documentación no contiene el diagrama que describe el paso a paso para resolver el requerimiento. O el diagrama no es claro o incorrecto. (resta 2 a la nota del entregable).	
			La documentación no contiene la explicación del patrón que permite resolver el requerimiento. O la explicación no es clara o incorrecta. (resta 2 a la nota del entregable).	

		La documentación no contiene la explicación de los nuevos componentes necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
		La documentación no actualiza el modelo de componentes que permita entender como se relacionan los componentes necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
		La documentación no actualiza el modelo de despliegue mostrando los componentes y los servicios necesarios para resolver el requerimiento. O estos no son claros o incorrectos. (resta 2 a la nota del entregable).	
		No se encuentra la documentación para resolver el requerimiento 7. O esta no es clara.	

	README	3	El archivo README fue actualizado con la nueva estructura del proyecto y el paso a paso de como desplegar todas sus aplicaciones en la nube.	
			El archivo README no fue actualizado con la nueva estructura del proyecto y el paso a paso de como desplegar todas sus aplicaciones en la nube.	
Implementación	Requerimiento RF006	33	El sistema permite crear y verificar tarjetas de crédito siguiendo el API definido y haciendo uso de la aplicación externa TrueNative. Los componentes se ejecutan y cumplen con el API definido para el requerimiento, ejecutándose de manera independiente y haciendo uso de Kubernetes o FaaS. Además, los componentes están descritos en los modelos y cumplen con las restricciones definidas para la entrega. La nota se calcula con base en el porcentaje de éxito del pipeline ci_evaluador_entrega3.yml. En el caso que el pipeline	

			haya sido modificado la nota será de 0.	
	Envío de correo electrónico RF006	7	El usuario que ejecuta la petición recibe el correo electrónico donde se presenta el estado final del proceso, el cuál es el mismo que está en la plataforma y puede ser verificado. Se calcula haciendo uso del pipeline ci_evaluador_entrega3_email.yml.	
			El usuario que ejecuta la petición no recibe el correo electrónico o su contenido no es correcto y consistente con el estado esperado en la plataforma. En el caso que el pipeline haya sido modificado la nota será de 0.	

	Requerimiento RF007	23	El sistema permite crear y verificar usuarios siguiendo el API definido y haciendo uso de la aplicación externa TrueNative. Los componentes se ejecutan y cumplen con el API definido para el requerimiento, ejecutándose de manera independiente y haciendo uso de Kubernetes o FaaS. Además, los componentes están descritos en los modelos y cumplen con las restricciones definidas para la entrega. La nota se calcula con base en el porcentaje de éxito del pipeline ci_evaluador_entrega3.yml. En el caso que el pipeline haya sido modificado la nota será de 0.	
	Envío de correo electrónico RF007	7	El usuario que ejecuta la petición recibe el correo electrónico donde se presenta el estado final del proceso, el cuál es el mismo que está en la plataforma y puede ser verificado. Se calcula haciendo uso del pipeline ci_evaluador_entrega3_email.yml.	

			El usuario que ejecuta la petición no recibe el correo electrónico o su contenido no es correcto y consistente con el estado esperado en la plataforma. En el caso que el pipeline haya sido modificado la nota será de 0.	
Pruebas	Pruebas unitarias de la aplicación de gestión de usuarios.	1	El código de gestión de usuarios tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el pipeline ci_evaluador_unit.yml.	
			El código de gestión de usuarios no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	
	Pruebas unitarias de la aplicación de gestión de publicaciones.	1	El código de gestión de publicaciones tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el pipeline ci_evaluador_unit.yml.	

			El código de gestión de publicaciones no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	
	Pruebas unitarias de la aplicación de gestión de ofertas.	1	El código de gestión de ofertas tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el pipeline ci_evaluador_unit.yml.	
			El código de gestión de ofertas no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	
	Pruebas unitarias de la aplicación de gestión de trayectos.	1	El código de gestión de trayectos tiene un cubrimiento de pruebas igual o superior al 70% y se valida correctamente por la ejecución de un job en el pipeline ci_evaluador_unit.yml.	

		El código de gestión de trayectos no tiene pruebas unitarias, no superan el 70%, o no existe un pipeline que ejecute correctamente estas pruebas.	

Requerimientos Funcionales

Estos son los nuevos requerimientos solicitados en el sistema.

Código	Historia
RF-006	Como usuario deseo almacenar mis tarjetas de crédito en la plataforma para poder realizar pagos de servicios si lo requiero.
RF-007	Como administrador del sistema deseo verificar la identidad de un usuario cuando es creado, de manera que pueda certificar que solo usuarios verificados pueden realizar publicaciones y ofertas en la plataforma.

RF-006

Código	RF-006
Historia	Como usuario deseo almacenar mis tarjetas de crédito en la plataforma para poder realizar pagos de servicios si lo requiero.

Criterios de aceptación	Cuando se solicita crear una tarjeta de crédito, nuestro sistema se debe comunicar con un sistema de pagos el cual generará un token de la tarjeta Solo los datos correspondientes a los últimos 4 dígitos de la tarjeta, la franquicia y un token generado por la plataforma de pagos se almacenan en el sistema, junto con el estado de la tarjeta POR_VERIFICAR Si la plataforma de pagos no puede verificar la tarjeta de crédito, el estado de la tarjeta debe ser RECHAZADA. Si la plataforma de pagos no puede generar el token de la tarjeta, esta no se almacena y muestra error al usuario. Si la plataforma de pagos logra verificar la tarjeta de crédito, esta queda almacena como VERIFICADA. Si el estado de la tarjeta es actualizada, la fecha de actualización de la tarjeta debe ser modificada usando la fecha actual. Indiferente del resultado del proceso, el usuario dueño de la tarjeta debe ser notificado por medio de correo electrónico el resultado de la inscripción de la tarjeta. El mensaje debe contener el resultado del proceso con los datos básicos de la tarjeta que está almacenada en la base de datos, y el RUV generado por el tercero.

RF-007

Código	RF-007
Historia	Como administrador del sistema deseo verificar la identidad de un usuario cuando es creado, de manera que pueda certificar que solo usuarios verificados pueden realizar publicaciones y ofertas en la plataforma.

Criterios de aceptación	Cuando el usuario es creado, el usuario se crea con estado POR_VERIFICAR. Mientras el usuario tenga el estado POR_VERIFICAR no podrá acceder al sistema. Nuestro sistema se comunica con un proveedor de verificación de identidad para verificar el usuario. El proveedor de identidad provee un puntaje de 1 a 100 sobre el análisis de identidad dependiendo de los datos brindados. Un puntaje igual o mayor de 60 indica que el usuario fue verificado, y menor, que no pudo serlo. La respuesta del proveedor debe actualizar el estado del usuario, el resultado podría ser VERIFICADO o NO_VERIFICADO. Si el estado del usuario es actualizado, la fecha de actualización del usuario debe ser modificada usando la fecha actual. Si el usuario mantiene el estado NO_VERIFICADO no podrá ingresar a la plataforma. Si el usuario queda en estado VERIFICADO, puede acceder a la plataforma para utilizar sus servicios. Indiferente del resultado del proceso, el usuario debe ser notificado por medio de correo electrónico del resultado del proceso. El correo electrónico debe contener el estado final del proceso, el RUV generado por el tercero, y los datos del usuario con el que se realizó el proceso de verificación.

Restricciones

Las siguientes restricciones se tienen que cumplir en la implementación. El incumplimiento de estas restricciones implica la no calificación de las entregas.

> Este documento puede estar siendo actualizado durante el desarrollo del curso, cualquier cambio será comunicado previamente a todos los estudiantes por los canales definidos.

Restricciones generales

Este proyecto se trabajará única y exclusivamente en los repositorios brindados por la dirección del curso. Si no ha sido agregado al repositorio, favor comuníquese con el tutor por medio de slack.
La estrategia para el proyecto es Monorepo. Haga uso de un solo repositorio de GitHub para almacenar el código fuente de todos los servicios. Cada componente debe estar en una carpeta aparte y no debe haber acoplamiento entre ellas.
El video de la entrega debe estar embebido dentro de la descripción del release, no puede estar publicado en otras plataformas ni se puede entregar por medio de enlaces. Si el video no se encuentra en la descripción del release, la nota de este será de cero.

Restricciones de la tercera entrega

Para la solución de los retos de esta entrega se debe hacer uso de los patrones y tecnologías vistas en el curso. Los patrones deben seguir las buenas prácticas de desacoplamiento y cohesión, lo que implica que está completamente prohibido el uso de Multiprocessing o Multithreading para la solución a los requerimientos de la entrega. Hacer uso de hilos o procesos para hacer polling o ejecutar tareas asíncronas sobre los componentes web de tarjetas o de usuarios conllevará a una nota de cero (0) en el requerimiento donde se evidencia esta práctica.
La entrega debe ejecutarse en el proveedor de nube seleccionado y será calificada solo sobre este ambiente, la ejecución exitosa en otros ambientes no es válido para su calificación.

Table of contents

API Tarjetas

Vista de información

Nuevas entidades en el sistema:

Entidad Tarjeta de Crédito

Campo	Tipo de dato	Descripción
id	cadena de caracteres	identificador de la tarjeta de crédito en formato uuid.
token	cadena de caracteres	token generado por la plataforma de pagos con una lóngitud de 256 caracteres.
userId	cadena de caracteres	identificador del usuario dueño de la tarjeta de crédito.
lastFourDigits	cadena de caracteres	últimos 4 dígitos de la tarjeta de crédito.
ruv	cadena de caracteres	registro único de verificación.
issuer	cadena de caracteres	Empresa que emite la tarjeta de crédito, y que puede ser: VISA, MASTERCARD, AMERICAN EXPRESS, DISCOVER, DINERS CLUB, UNKNOWN.

status	cadena de caracteres	Estado de la tarjeta, solo una de estas opciones es válida: POR_VERIFICAR, RECHAZADA, APROBADA
createdAt	datetime	fecha y hora de creación de la tarjeta de crédito.
updatedAt	datetime	fecha y hora de actualización de la tarjeta de crédito.

API Tarjetas

El API de tarjetas de crédito permite crear y consultar tarjetas de crédito.

la siguiente especificación del API es obligatoria, pero no impide que ustedes como equipo puedan agregar más endpoints si lo requieren por otros motivos o propósitos.

1. Creación de tarjetas

Crea una tarjeta asociada al usuario al que pertenece el token.

Método	POST
Ruta	/credit-cards
Parámetros	

Encabezados	`Authorization: Bearer token`
Cuerpo	`{ "cardNumber": Número de la tarjeta (cadena de caracteres con todos sus caracteres numéricos y sin espacios), "cvv": Número de verificación de la tarjeta, "expirationDate": Fecha de expiración de la tarjeta en el formato YY/MM, "cardHolderName": Nombre del propietario grabado en la tarjeta }`

Respuestas

Código	Descripción	Cuerpo
401	El token no es válido o está vencido.	
403	No hay token en la solicitud	
400	En el caso que alguno de los campos no esté presente en la solicitud, no tengan el	

	formato esperado o lleguen vacíos.	
409	Si el usuario ya había almacenado la misma tarjeta de crédito.	
412	Si la tarjeta de crédito ya está vencida.	
201	En el caso que la tarjeta se haya creado con éxito.	{ "id": id de la tarjeta, "userId": id del usuario que creo la tarjeta, "createdAt": fecha y hora de creación de la tarjeta en formato ISO }

2. Ver tarjetas

Retorna el listado de tarjetas de crédito que pertenecen con el usuario que está realizando la solicitud.

Método	GET

Ruta	/credit-cards
Parámetros	
Encabezados	`Authorization: Bearer token`
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
401	El token no es válido o está vencido.	
403	No hay token en la solicitud	

200	Listado de tarjetas de crédito que pertenecen al usuario que realizó la solicitud.	

El token de la tarjeta NO debería ser retornado como parte de la respuesta de nuestros servicios. Para este proyecto se incluye como parte de la respuesta por propósito educativo.

3. Consultar cantidad de entidades

Usado para consultar cuantas entidades de tarjetas están en la base de datos.

Método	GET
Ruta	/credit-cards/count
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Cantidad de entidades almacenadas en la base de datos.	{"count": número positivo o cero}

4. Consulta de salud del servicio

Usado para verificar el estado del servicio.

Método	GET
Ruta	/credit-cards/ping
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo

200	Solo para confirmar que el servicio está arriba.	`pong`

5. Restablecer base de datos

Usado para eliminar todas las tarjetas.

Método	POST
Ruta	/credit-cards/reset
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Todos los datos fueron eliminados.	`{"msg": "Todos los datos fueron eliminados"}`

Recursos para la entrega 3

Código

> Haga uso de los recursos de la Entrega 1 y Entrega 2 para la construcción de aplicaciones.
> Los talleres vistos durante estas semanas le enseñan cómo desplegar en la nube usando Terraform y K8s.

> Le recomendamos crear un archivo de scripts para desplegar siempre su infraestructura de manera replicable. Por ejemplo, usar `make`.

Pruebas

Para probar sus aplicaciones localmente y validar que responden correctamente a la especificación de API REST, puede usar las siguientes colecciones. Descargue cada archivo JSON e impórtelo en su aplicación de Postman. Estas mismas pruebas son usadas para la calificación de su entrega.

> Pruebas entrega 3:
> https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/recursos-evaluador/main/entreg3/entrega3.json
> Pruebas entrega 3 con correo personalizado:
> https://raw.githubusercontent.com/MISW-4301-Desarrollo-Apps-en-la-Nube/recursos-evaluador/main/entreg3/entrega3_monitor.json

Para importar la colección use el botón Import en la sección superior izquierda.

Una vez importada la colección, actualice la variable de ambiente de la ruta de la aplicación que desea probar. No modifique el `Initial value` (valor inicial), modifique el `Current value` (valor actual).

Finalmente, ejecute la colección haciendo clic derecho en el nombre de la colección y seleccionando `Run`. Si la ruta de la aplicación está correctamente configurada Postman ejecutará múltiples solicitudes de API y también ejecutará algunos assertions que hemos preparado para asegurarnos de que la aplicación esté funcionando como se espera.

Pipelines

En su repositorio de equipo encontrará una serie de pipelines para verificar que su entrega cumple con parte de los entregables calificables. Estos pipelines se ejecutan como un workflow de Github Actions de forma manual o automática dependiendo del workflow.

El workflow que revisa la documentación y las pruebas unitarias se ejecuta cada que usted mezcle en la rama `main` o cuando haga un Pull Request a la rama `main`. El workflow que ejecuta las pruebas de k8s solo se ejecuta manualmente y lo puede ejecutar en cualquier rama. Para ello diríjase a la sección de `Actions` del repositorio.

Encontrará en el menú izquierdo una lista de todos los flujos de trabajo (workflows) disponibles para ejecución. En este caso, verá `Evaluador Implementación Entrega 3` que corresponde al pipeline de pruebas de sus aplicaciones. Haga clic en el workflow y verá un botón `Run workflow` en la sección superior derecha. Haga clic en este botón, seleccione la rama en la que desea ejecutarlo y por último oprima el botón `Run workflow`.

Esto iniciará la ejecución del workflow en la rama. Si todo funciona correctamente y la entrega es correcta, verás que todas las comprobaciones aparecen como aprobadas (passed).

ecnologías del proyecto

El equipo de desarrollo podrá seleccionar los lenguajes con los que desarrollarán las aplicaciones, sin embargo, las tecnologías de ejecución y despliegue, así como las tecnologías de bases de datos, no serán opcionales y solo podrán usarse las ya definidas.

En esta entrega continuaremos usando las tecnologías descritas en la Entrega 1 y la Entrega 2 y se agregan las siguientes como opcionales:

Tecnologías opcionales desde la tercera entrega

- DynamoDB
- SQS
- SNS
- Lambda
- Knative

> Nota: Les recomendamos estar vigilando los créditos disponibles en su cuenta con el fin de evitar un consumo total antes de realizar la entrega del proyecto.

Table of contents

TrueNative

TABLE OF CONTENTS

En este documento se describe como se debe hacer uso de la aplicación TrueNative la cual simula ser un verificador de identidad y un gestor de registro de tarjetas de crédito, que puede configurar y ejecutar directamente en su cluster. Tenga presente las restricciones y configuraciones que debe cumplir para su ejecución.

Para hacer uso de este servicio en su cluster utilice la imagen ghcr.io/misw-4301-desarrollo-apps-en-la-nube/true-native:2.0.0 la cual la encuentra en el registro de paquetes de esta organización.

Adicionalmente podrá encontrar un repositorio con algunos archivos de ejemplo para desplegar en k8s y colecciones de postman para que pueda probar la aplicación localmente.

Haga uso de la URI de la imagen en su configuración de Kubernetes para hacer el despliegue.

Configuración

La aplicación debe ser configurada con las siguientes variables de ambiente de manera obligatoria:

Variable	**Tipo**	**Descripción**
SECRET_TOKEN	cadena de caracteres random	Token que debe ser enviado en cada petición por parte de las aplicaciones cliente, con el fin de certificar que la petición la realiza una aplicación certificada. (Cualquier cadena de caracteres definida por usted)

Al tiempo la aplicación tiene otras variables de ambiente opcionales que pueden ser utilizadas para facilitar las pruebas.

Variable	Tipo	Descripción
MAX_WEBHOOK_DELAY	número positivo mayor a 10 y menor a 120, default 120	Indica el máximo de segundos posteriores en que el sistema responderá el resultado de la verificación de usuario al endpoint definido. El objetivo de esta variable es simular la espera de una plataforma real y así pueda realizar pruebas más sencillas.
MAX_POLL_DELAY	número positivo mayor a 10 y menor a 30, default 30	Indica los segundos posteriores en que el sistema responderá el resultado de la verificación. El objetivo de esta variable es simular la espera de una plataforma real y así pueda realizar pruebas más sencillas.
SUCCESS_RATE	número positivo entre 1 y 100, default 50	Indica el porcentage de éxito en la verificación de tarjeta de crédito o del usuario. El objetivo de esta variable es simular el comportamiento real de verificación, puede manipularse para realizar pruebas más sencillas. Por ejemplo, si el valor es definido como 100, todas las solicitudes serán aprobadas/verificadas, si el valor es definido como 0, todas las solicitudes serán marcadas como no verificadas/rechazadas.

Funcionamiento

En las siguientes páginas puede encontrar el detalle de como funcionan cada uno de los servicios que ofrece TrueNative:

Servicio	Link
Verificación de identidad de usuario	Link
Registro y verificación de tarjetas de crédito	Link

Endpoints de gestión

1. Consulta de salud del servicio

Usado para verificar el estado del servicio.

Método	GET
Ruta	/native/ping
Parámetros	
Encabezados	
Cuerpo	

Respuestas

Código	Descripción	Cuerpo
200	Solo para confirmar que el servicio está arriba.	pong

Verificación de identidad de usuario

TABLE OF CONTENTS

El servicio de verificación de usuario se encarga de usar la información provista de un individuo y compararla con datos en varias fuentes de información ya recolectadas. La plataforma TrueNative se compromete a tener la respuesta de la verificación en no más de 120 segundos y comunicar la respuesta por medio de un endpoint provisto en su plataforma.

Para solicitar una verificación haga uso del siguiente endpoint.

Crear solicitud de verificación de usuario

Método	POST
Ruta	/native/verify
Parámetros	
Encabezados	`Authorization: Bearer secret_token`

Cuerpo	`{`
	`  "user": {`
	`  "email": email del usuario a verificar,`
	`  "dni": opcional,`
	`  "fullName": opcional,`
	`  "phone": opcional`
	`  },`
	`  "transactionIdentifier": Valor único definido por el cliente, usado para hacer trazabilidad de la petición. Cada petición debe tener uno distinto.`
	`  "userIdentifier": Valor que identifica el usuario en sus sistemas.`
	`  "userWebhook": url del webhook del usuario donde la aplicación notificará el resultado de la verificación en el caso que la solicitud sea ejecutada en el modelo básico.`
	`}`

Token de autenticación	Usando el token secreto SECRET_TOKEN definido como variable de ambiente.

Objeto user	El objeto user contiene la información del usuario a verificar, entre más datos contenga este objeto mayor la probabilidad de éxito en la verificación. Los campos que puede contener son "email", "dni", "nombre completo", "teléfono".
Webhook	Url donde la aplicación TrueNative puede notificar la respuesta de la verificación después de los segundos definidos en la variable WEBHOOK_DELAY. Ejemplo: users/40

Las posibles respuestas a esta petición son:

Respuestas

Código	Descripción	Cuerpo
400	En el caso que el objeto de usuario no esté en la petición o esté vacío, falte el userIdentifier, falte el transactionIdentifier, falte el webhook, o esté malformada la petición.	
401	El secret token no corresponde	

	al definido en el servicio.	
403	La solicitud no incluya el secret token en los encabezados.	
409	Si ya hay una solicitud de verificación en ejecución para el identificador de usuario e identificador de la transacción.	

201	Si la solicitud de verificación ha sido creada correctamente.	{
		"RUV": registro único de verificación,
		"userIdentifier": Valor que identifica el usuario en sus sistemas,
		"transactionIdentifier": Valor único definido por el cliente
		para hacer trazabilidad de la petición.
		Cada petición debe tener uno distinto,
		"createdAt": fecha de creación de la solicitud,
		"task_status": ACCEPTED
		}

Callback

La respuesta de la verificación del usuario será publicada al webhook definido en la solicitud. Este endpoint debe ser público, procesar la acción PATCH y responder el código de estado 200; TrueNative intentará realizar una petición PATCH con el siguiente cuerpo al endpoint definido:

```
{
    "RUV": registro único de verificación,
    "userIdentifier": identificador definido por el consumidor
    "createdAt": fecha de creación de la solicitud,
    "status": NO_VERIFICADO o VERIFICADO,
    "score": valor numérico de 0 a 100 que indica que tan seguro es el resultado,
    "verifyToken": token con el que puede verificar que la respuesta es válida
     (proviene de TrueNative y no ha sido alterada en la comunicación).
}
```

Tenga presente que un usuario que recibe un score inferior o igual a 60 responderá con el status NO_VERIFICADO, y VERIFICADO si el score dado al usuario supere el umbral de 60. Entre menos datos del usuario envíe, la probabilidad de éxito será menor, dado que no hay suficiente información para verificar el usuario.

Dado que el webhook brindado para la respuesta es público (no hay mecanismo de autorización) puede verificar que el mensaje no ha sido alterado generando un SHA256 de los campos SECRET_TOKEN, RUV y SCORE. Un código de ejemplo para hacerlo sería:

```
import hashlib

...

token = f"{SECRET_TOKEN}:{RUV}:{SCORE}"

sha_token = hashlib.sha256(token.encode()).hexdigest()

assert sha_token == response["verifyToken"]
```

De esta forma se asegura que la respuesta que llega al webhook proviene de un cliente que usa el mismo token secreto definido por usted.

Endpoints de auditoría

Los siguientes endpoints están diseñados para facilitar sus pruebas y desarrollo, no tienen un propósito funcional y NO deben ser usados en el flujo que resuelve el requerimiento funcional.

Listado de solicitudes

El siguiente endpoint tiene como propósito mostrar el estado de sus solicitudes de verificación en el sistema.

Método	GET
Ruta	native/verify/log
Parámetros	
Encabezados	
Cuerpo	

Las posibles respuestas a esta petición son:

Respuestas

Código	Descripción	Cuerpo

200	Si el estado es PENDING indica que la verificación está siendo procesada, si el estado es PROCESSED el resultado de la verificación ya fue entregado (o se intentó entregar) al webhook definido.	[{ "RUV": registro único de verificación, "userIdentifier": identificador definido por el consumidor "createdAt": fecha de creación de la solicitud, "task_status": estado actual de la transacción, puede ser PENDING|PROCESSED }]

Detalle de una solicitud

El siguiente endpoint tiene como propósito mostrar estado de una solicitud con el listado de eventos que ha tenido.

Método	GET
Ruta	native/verify/log/{ruv}
Parámetros	
Encabezados	

Cuerpo	

Las posibles respuestas a esta petición son:

Respuestas

Código	Descripción	Cuerpo
404	Si no existe una transacción con el RUV provisto	

200	Si el estado es PENDING indica que la verificación está siendo procesada, si el estado es PROCESSED el resultado de la verificación ya fue entregado (o se intentó entregar) al webhook definido.	{
		"RUV": registro único de verificación,
		"userIdentifier": identificador definido por el consumidor
		"createdAt": fecha de creación de la solicitud,
		"task_status": estado actual de la transacción,
		puede ser PENDING|PROCESSED,
		"events": [
		describe los eventos que ha tenido una solicitud en la aplicación
		]
		}

Limpieza de solicitudes

El siguiente endpoint tiene como propósito eliminar todas las solicitudes de verificación y registros de auditoría de la aplicación.

Método	DELETE
Ruta	native/verify/log
Parámetros	
Encabezados	
Cuerpo	

Las posibles respuestas a esta petición son:

Respuestas

Código	Descripción	Cuerpo
200		Todas las solicitudes de verificación fueron eliminadas

Registro y verificación de tarjetas de crédito

TABLE OF CONTENTS

El servicio de verificación de tarjetas de crédito se encarga de usar la información provista de una tarjeta de crédito y realizar un pago mínimo para verificar que la tarjeta está activa. La compañía TrueNative se compromete a tener la respuesta de la verificación en no más de 30 segundos.

Para registrar una tarjeta haga uso del siguiente endpoint.

Registrar tarjeta

Método	POST
Ruta	/native/cards
Parámetros	

Encabezados	`Authorization: Bearer secret_token`

Cuerpo	`{`
	`  "card": {`
	`  "cardNumber": Número de la tarjeta (cadena de caracteres con todos sus caracteres numéricos y sin espacios),`
	`  "cvv": Número de verificación de la tarjeta (cadena de caracteres),`
	`  "expirationDate": Fecha de expiración de la tarjeta en el formato YY/MM,`
	`  "cardHolderName": Nombre del propietario grabado en la tarjeta`
	`  },`
	`  "transactionIdentifier": Valor único definido por el cliente, usado para hacer trazabilidad de la petición.`
	`  Cada petición debe tener uno distinto.`
	`}`

Token de autenticación	Usando el token secreto SECRET_TOKEN definido como variable de ambiente.
Objeto card	El objeto card contiene la información de la tarjeta a registrar, todos los campos son obligatorios.

Las posibles respuestas a esta petición son:

Respuestas

Código	Descripción	Cuerpo

400	En el caso que el objeto de tarjeta no esté en la petición o esté vacío, falte el transactionIdentifier, o esté malformada la petición.	
401	El secret token no corresponde al definido en el servicio.	
403	La solicitud no incluye el secret token en los encabezados.	
409	Si ya hay una solicitud de verificación en ejecución para la tarjeta.	

201	Si la solicitud de verificación ha sido creada correctamente.	{
		"RUV": registro único de verificación,
		"token": Token de la tarjeta de crédito,
		"issuer": *Nombre del emisor de la tarjeta,
		"transactionIdentifier": Valor único definido por el cliente
		para hacer trazabilidad de la petición.
		Cada petición debe tener uno distinto,
		"createdAt": fecha de creación de la solicitud
		}

> Si la solicitud de verificación fue creada. Se debe hacer uso de polling en el endpoint de native/cards/{ruv} para conocer su siguiente estado.

Verificar estado de la solicitud

Método	GET
Ruta	/native/cards/{ruv}
Parámetros	
Encabezados	`Authorization: Bearer secret_token`
Cuerpo	
Token de autenticación	Usando el token secreto SECRET_TOKEN definido como variable de ambiente.

Las posibles respuestas a esta petición son:

Respuestas

Código	Descripción	Cuerpo
404	No existe una verificación en proceso con ese RUV.	
401	El secret token no corresponde al definido en el servicio.	
403	La solicitud no incluya el secret token en los encabezados.	
202	El proceso de registro y verificación aún está en proceso.	

200	Si la verificación fue realizada con éxito.	{
		"RUV": registro único de verificación,
		"createdAt": fecha de creación de la solicitud,
		"transactionIdentifier": Valor único definido por el cliente,
		"status": APROBADA o RECHAZADA
		}

Endpoints de auditoria

Los siguientes endpoints están diseñados para facilitar sus pruebas y desarrollo, no tienen un propósito funcional y NO deben ser usados en el flujo que resuelve el requerimiento funcional.

Listado de solicitudes

El siguiente endpoint tiene como propósito mostrar el estado de sus solicitudes de verificación en el sistema.

Método	GET
Ruta	native/cards/log
Parámetros	
Encabezados	
Cuerpo	

Las posibles respuestas a esta petición son:

Respuestas

Código	Descripción	Cuerpo
200	Si el estado es PENDING indica que la verificación está siendo procesa, si el estado es PROCESSED el resultado de la verificación ya fue entregado.	[{ "RUV": registro único de verificación, "token": token de la tarjeta, "transactionIdentifier": Valor único definido por el cliente "createdAt": fecha de creación de la solicitud, "status": estado actual de la transacción, puede ser PENDING|PROCESSED }]

Detalle de una solicitud

El siguiente endpoint tiene como propósito mostrar el estado de una solicitud con el listado de eventos que ha tenido.

Método	GET
Ruta	native/cards/log/{ruv}
Parámetros	
Encabezados	

Cuerpo	N/A

Las posibles respuestas a esta petición son:

Respuestas

Código	Descripción	Cuerpo
404	Si no existe una transacción con el RUV provisto	

200	Si el estado es PENDING indica que la verificación está siendo procesa, si el estado es PROCESSED el resultado de la verificación ya fue entregado.	<pre>{
		"RUV": registro único de verificación,
		"token": token de la tarjeta,
		"transactionIdentifier": Valor único definido por el cliente
		"createdAt": fecha de creación de la solicitud,
		"status": estado actual de la transacción,
		puede ser PENDING|PROCESSED,
		"events": [
		describe los eventos que ha tenido una solicitud en la aplicación
		]
		}</pre>

Limpieza de solicitudes

El siguiente endpoint tiene como propósito eliminar todas las solicitudes de verificación y registros de auditoría de la aplicación.

Método	DELETE
Ruta	native/cards/log
Parámetros	
Encabezados	
Cuerpo	

Las posibles respuestas a esta petición son:

Respuestas

Código	Descripción	Cuerpo
200		Todas las solicitudes de verificación fueron eliminadas