Arquitectura de Software en la Práctica
Obligatorio 1

Objetivos

Desarrollar los conceptos que definen la computación en la nube mediante la adopción de técnicas de construcción de software como servicio, desde el desarrollo hasta su despliegue en un entorno de producción.

Problema

Luego de finalizar la universidad usted y dos de sus compañeros de carrera deciden comenzar su propio emprendimiento. La idea surge a partir de una necesidad de la ex-compañía de uno de sus compañeros.

La necesidad en cuestión es la creación, gestión y trazabilidad de cupones promocionales y descuentos, pudiendo ser gestionado desde un mismo lugar. Es decir, dado un cupón que una compañía ofrezca el sistema deberá validar si el cupón es válido y si además cumple con las reglas para que sea aplicado.

Ejemplos:

“Obtené un cupón de 5% de descuento para tu primera compra en las cantinas de la Universidad ORT ingresando el código COMIDADESC5”

“Con tu primera compra en Mochileros.uy un cupón de $150 de descuento para tu próxima compra!”

“Los usuarios que ingresen el cupón COMPRAYA obtienen un 10% de descuento en su compra de Uniformes.uy”

A su vez, el sistema también deberá soportar aplicar descuentos en tiempo real dadas ciertas condiciones en el proceso de compra de una plataforma online, como si la compra supera cierto monto, y cantidad de ítems entonces se aplica el descuento.

Ejemplos:

“Compras mayores de $1000 obtienes un 10% de descuento”

“Comprando 3 unidades de nuestras mochilas escolares te descontamos $300”

Además, es posible combinar ambas soluciones al mismo tiempo, como por ejemplo, tener un cupón que aplique si y sólo si el monto es superior a un número determinado.

Luego de realizar una investigación descubre que es una necesidad común en el mundo del e-commerce. Es por eso que en una primera etapa logró cerrar acuerdos para la construcción de una versión simplificada del producto con cinco clientes.

Se decide ofrecer este software como servicio. Es entonces que usted y su equipo se ponen como objetivo revolucionar el mercado ofreciendo un producto fácil de usar a través de un sistema web moderno. Para esto, comienza a desarrollar *Coupons* que pretende ser el servicio de cupones y descuentos referente de la región.

Como este producto es ofrecido a diferentes clientes entonces debe tener cuidado en cómo almacena sus datos y que por ejemplo los datos de una empresa *A* no se visualicen o interfieran en o con los datos de una empresa *B*.

Desarrollo

El proyecto tecnológico a desarrollar se deberá ajustar a las características de una aplicación *cloud native*. Sin embargo, dada la escasa familiaridad del equipo con tecnologías *cloud* se deberán mantener los esfuerzos iniciales de *DevOps* y orquestación de servicios a un mínimo, motivo por el cual se decide utilizar un servicio *PaaS* que permita facilitar las iteraciones rápidas y automatizadas.

Requerimientos funcionales

RF1. Registro de usuario

Actor: Usuario no registrado
Los usuarios finales podrán registrarse de dos maneras:

1. Registro vía web: Se deberá ofrecer una página web que permita a cualquier persona inscribirse como usuario. Los datos a ingresar son: nombre, apellido, correo electrónico, organización a la que pertenece y foto de perfil (opcional). Por defecto esta persona queda como administradora de la organización.
2. Registro vía link de invitación: Un usuario administrador de una organización puede invitar a otros miembros de su organización no registrados en el sistema. Para ello, podrá ingresar un correo, al que se le enviará un correo electrónico que contenga un link al registro anterior (1.) con la organización precargada y no editable.

RF2. Autenticación de usuario

Actor: Usuario administrador y usuario de organización

Dado un usuario no autenticado cuando esta presenta en la página principal del sistema y hace click en "Ingresar" entonces se despliega un formulario de ingreso que solicita su correo electrónico y un contraseña de acceso. Una vez autenticado el usuario el sistema debe redireccionarlo a la pantalla de listado de promociones.

RF3. Gestión de clave de aplicación

Actor: Usuario administrador

El sistema debe permitir a los usuarios administradores gestionar claves de acceso de aplicación. Estas claves únicas de acceso son las que el cliente debe utilizar para invocar los servicios provistos por el sistema. Ver RNF 4

De las claves de acceso simplemente se solicita un nombre para identificarlas y qué promociones están asociadas a la misma. Ver RF 5

RF4. Gestión de promociones

Actor: Usuario administrador

Se debe disponibilizar la gestión de promociones por cupones, de cada promoción de este tipo se conoce: su código, su nombre, el tipo y valor de retorno (porcentaje o valor fijo), si está activa o no y el tipo de la promoción (cupón o descuento).

Se debe especificar la lista de atributos que serán evaluados para determinar si se tiene que aplicar la promoción, estos atributos deben ser almacenados como *metadata* de la promoción. El atributo *coupon_code* es mandatorio en el tipo de promoción cupón, pues es necesario validar que un cupón se otorgue una sola vez, en el caso de las promociones por descuento es mandatorio contar con el atributo *transaction_id* (*id* de la transacción en el sistema externo) por el mismo motivo de validación.

Adicionalmente, se deben especificar las condiciones por medio de estructuras condicionales pre-definidas (IF, AND, OR, THEN), operadores predefinidos (<,>,<=,>=,=) y los atributos anteriormente mencionados.

Ejemplo promoción tipo cupón:

- Nombre: "Promo verano"
- Retorno: "10%" (tipo de descuento "porcentaje" con valor 10)
- Activa: "Si"
- Tipo promocion: "Cupón"
- Atributos: *coupon_code*, *total*, *products_size*

- Condiciones:
    - IF *valid_coupon_code* (1)
    - AND *total > 100*
    - AND *products_size* >=2
    - THEN *apply_discount* (2)

1. Verifica que el *coupon_code* sea válido y no haya sido canjeado previamente. Esta condición es mandatoria.
2. Si cumple con la condición retorna que se puede aplicar la promoción y retorna el tipo de descuento y valor del mismo.

Ejemplo promoción tipo descuento:

- Nombre: "Descuento comienzo clases"
- Retorno: "$100" (tipo de descuento "valor" con valor 10)
- Activa: "Si"
- Tipo promocion: "Descuento"
- Atributos: *transaction_id, total*, *quantity*
- Condiciones:
    - IF *valid_transaction* (1)
    - OR (*total <= 1000 AND quantity* >= 5*) (2)*
    - OR *total* > 1000
    - THEN *apply_discount* (3)

1. Verifica que el *transaction_id* no haya sido canjeado previamente para una promoción. Esta condición es mandatoria.
2. Se pueden anidar condiciones
3. Si cumple con la condición retorna que se puede aplicar la promoción y retorna el tipo de descuento y valor del mismo.

Una vez dada de alta la promoción esta es almacenada en el sistema para posteriormente ser consultada.

RF5. Listado de promociones

Actor: Usuario administrador y usuario de organización

Se debe disponibilizar un listado de promociones, permitiendo filtrar por código, nombre, estado y tipo de promoción.

En el listado se debe desplegar el código de la promoción, su nombre, su estado y tipo, además de enlaces las operaciones de modificación, baja y reporte.

RF6. Baja de promocion

Actor: Usuario administrador

Se deberá permitir dar de baja una promoción. En caso de baja, toda consulta por dicha promoción deberá responder como "no aplica", es decir, que no cumple con las condiciones necesarias para aplicar la promoción. No se debe bajo ningún concepto eliminar físicamente la promoción, ya que debe seguir apareciendo en los reportes a efectos estadísticos.

RF7. Reporte de uso

Actor: Usuario administrador y usuario de organización

Se debe disponibilizar la opción de realizar consultas sobre una promoción. Este reporte deberá mostrar los siguientes datos:

- Cantidad de veces que ha sido invocado una promoción
- Tasa de respuesta positiva / negativa
- Tiempo promedio de respuesta en evaluar una promoción
- Suma del dinero gastado por la promoción

Adicionalmente, esta funcionalidad debe estar expuesta como un endpoint *REST* que debe responder en promedio por debajo de los **100 ms** para cargas de hasta **1200 *req/m*,** ya que va a ser utilizado en un futuro por distintos *backoffice* de los clientes.

RF8. Evaluación de promoción

Actor: Sistema externo

Se deberá disponibilizar un endpoint *REST* que permita consultar a un sistema si determinada promoción aplica o no y brindar la información necesaria para la toma de acciones.

Dicha consulta se debe realizar por el código de la promoción especificando los atributos requeridos por la misma.

El sistema debe retornar si la promoción aplica o no para los atributos especificados, en caso de que aplique se debe especificar cual es el tipo de descuento a aplicar (porcentaje o valor) y el monto del mismo.

Requerimientos no funcionales

RNF1. Performance

El tiempo de respuesta promedio para todas las operaciones públicas del sistema en condición de funcionamiento normal deberá mantenerse en promedio por debajo de los **200 ms.** para cargas de hasta **1200 *req/m***.

RNF2. Confiabilidad y disponibilidad

Con el fin de poder monitorear la salud y disponibilidad del sistema, se deberá proveer un *endpoint* HTTP de acceso público que informe el correcto funcionamiento del sistema (conectividad con bases de datos, colas de mensajes, disponibilidad para recibir requests, etc.). El mismo deberá responder a `GET /healthcheck` con *status* `200 OK`, siendo cualquier otro código de respuesta considerando como una disrupción en el servicio. No se requieren formatos particulares para el cuerpo de la respuesta.

RNF3. Configuración y manejo de secretos

Relativo al desarrollo, todo dato de configuración sensible que se maneje en el código fuente deberá poder especificarse en tiempo de ejecución mediante variables de entorno, manteniendo todo valor fuera del repositorio. Esto incluye credenciales de acceso a *APIs*, *URLs* de servicios externos, y cualquier otra configuración específica del sistema.

RNF4. Autenticación y autorización

Se deberá establecer un control de acceso basado en roles, distinguiendo entre usuarios administradores y usuarios de una organización. Los permisos otorgados a los mismos deberán restringir el acceso a sus correspondientes funcionalidades, prohibiendo la interacción con cualquier otra operación pública del sistema.

Las claves de aplicación deben ser generadas como *JSON Web Tokens* y el sistema debe validar que el *token* sea válido y tenga los permisos necesarios para acceder a los endpoints especificados en RF 7 y RF 8.

RNF5. Seguridad

El sistema deberá responder con código `40X` a cualquier *request* mal formada o no reconocida por el mismo, no dejando expuesto ningún *endpoint* que no sea explícitamente requerido por alguna funcionalidad.

A su vez, toda comunicación entre clientes *front end* y componentes de *back end* deberán utilizar un protocolo de transporte seguro. Por otra parte, la comunicación entre componentes de *back end* deberá en lo posible realizarse dentro de una red de alcance privado; de lo contrario deberán también utilizar un protocolo de transporte seguro autenticados por una clave de autenticación.

RNF6. Código fuente

El sistema deberá ser desarrollado con el lenguaje Ruby, utilizando el framework Ruby on Rails para las funcionalidades de acceso web. Para mejorar la mantenibilidad del proyecto, todo el código y comentarios deberán ser escritos en inglés, siguiendo la guía de estilos de Ruby y verificado con Rubocop.

Por otra parte, el control de versiones del código se deberá llevar a cabo con repositorios Git debidamente documentados, que contengan en el archivo `README.md` una descripción con el propósito y alcance del proyecto, así como instrucciones para configurar un nuevo ambiente de desarrollo. Para el manejo de branches, se deberá utilizar Gitflow.

RNF7. Pruebas

Se deberá mantener un *script* de generación de planes de prueba de carga utilizando la herramienta Apache jMeter, con el objetivo de ejercitar todo el sistema simulando la actividad de múltiples usuarios concurrentes. De necesitar incluir claves o secretos, el *script* deberá recibir dichas configuraciones por variables de entorno.

Además de las pruebas de carga, se deberá contar con pruebas automatizadas para los distintos requerimientos funcionales.

RNF8. Identificación de fallas

Para facilitar la detección e identificación de fallas, se deberán centralizar y retener los *logs* emitidos por la aplicación en producción por un período mínimo de 24 horas.

Documentación

Además de la solución a implementar, se pide incluir una documentación con los siguientes puntos:

Descripción de la arquitectura

Deberá mostrar las partes que componen al sistema a través de vistas de módulos, componentes y conectores, y despliegue. Deberá asegurarse de mostrar:

- Distribución de todo el sistema en componentes físicos.
- Flujo de información a través de la infraestructura en tiempo de ejecución.

Justificaciones de diseño

Se pide un detalle de las tácticas aplicadas para conseguir los requerimientos no funcionales exigidos por la especificación. Las mismas pueden incluir tanto decisiones explícitamente introducidas en su diseño (esto es, en *userland*), como aquellas contenidas en las soluciones ya ofrecidas por la plataforma o *framework* utilizados.

Descripción del proceso de *deployment*

Se deberán describir los pasos que comprenden el pasaje del sistema en su entorno de desarrollo al entorno de producción (compilación de *assets*, migraciones, ejecución de pruebas, etc.), incluyendo un detalle de las tácticas utilizadas para preservar la disponibilidad del servicio durante este proceso y cómo fueron implementadas.

Entrega

La entrega podrá realizarse hasta el día **10 de octubre** a las **21 horas** a través de gestion.ort.edu.uy. La entrega incluye:

- URI para el acceso web público a una instancia en producción del sistema *Coupons*.
- Dirección de los repositorios privados en Github.
- Cualquier conjunto de credenciales adicionales que sean necesarias para el uso de *Coupons*.