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

Usted es el dueño de una empresa de desarrollo de software, varios de sus clientes le han trasladado inquietudes a la hora de desplegar nuevas funcionalidades a sus respectivos usuarios. Las necesidades que le trasmiten son: desplegar una funcionalidad a un cierto número de usuarios, a un porcentaje de ellos o simplemente mostrar variaciones de una funcionalidad a un grupo de usuarios con la finalidad de verificar cual es mejor.

Luego de investigar en la web descubre que M. Fowler describe una técnica para solucionar estos inconvenientes: *Feature Toggles* o *Feature Flags*. Esta técnica permite cambiar el comportamiento de un sistema sin cambiar el código. Básicamente se definen "flags" que actúan como "if" y dado la solicitud de un usuario retorna "verdadero" o "falso" que luego es utilizado para variar el comportamiento del sistema. Por ejemplo: existe un "flag" llamado "Mostrar botón rojo" que aplica al 30% de los usuarios, entonces el sistema debe permitir consultar si determinados usuarios están o no dentro de ese 30% para decidir si el botón debe desplegarse en color rojo o en el color usual.

Dado que la mayoría de sus clientes le trasladaron esta inquietud usted realizó un estudio para analizar la viabilidad de ofrecer este software como servicio. Es entonces que luego de obtener respuestas alentadoras quiere revolucionar el mercado ofreciendo un servicio fácil de usar a través de un sistema web moderno. Para esto, comienza a desarrollar *Flags* que pretende ser el servicio de *Feature Toggles* referente de la región.

Supongamos que la empresa *A* quiere desplegar una nueva funcionalidad a sus usuarios, en este caso, esa nueva funcionalidad tiene un impacto grande en la usabilidad por lo que manifiestan temor si es liberado a todos sus usuarios, el sistema debe permitir en estos casos liberar a una cantidad porcentual de usuarios. Si un usuario ya tiene disponible la nueva funcionalidad entonces debe tenerla siempre a no ser que esa funcionalidad se coloque como "apagada". Por otra parte la empresa *B* liberó una nueva funcionalidad que potencialmente tiene un impacto negativo en la performance del sistema, por lo que si bien quieren probarla para todos sus usuarios el sistema les debe permitir "apagar" y "encender" dicha funcionalidad en tiempo real sin realizar cambios en su código.

Como *Flags* es un servicio ofrecido a muchos clientes entonces debe tener cuidado en cómo almacena sus datos y que por ejemplo los "flags" de la empresa *A* no se visualicen o interfieran en o con los "flags" de la 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). Es deseable que se puedan obtener estos datos a partir de un servicio autenticador externo (Google, Facebook, Twitter, etc). 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 "flags".

RF3. Registro de "flag"

Actor: Usuario administrador

Se debe disponibilizar el registro de "flags", de cada flag se conoce: su nombre, el tipo (porcentaje, lista fija o booleano) y si está activa o no. Luego de acuerdo el tipo se solicita información adicional:

- *Booelana:* es el caso más sencillo, si el flag está activado retorna el valor preestablecido "verdadero" o "falso"
- *Porcentaje:* se especifica el porcentaje de usuarios que debe ser afectado cuando la "flag" se encuentra activa. Ejemplo: 10%, 30%, 50%, etc.
- *Lista de usuarios:* cuando se solicita evaluar un flag los sistemas externos deben enviar información contextual del usuario que realiza la acción, esta información debe ser especificada dentro de una lista de valores predefinidas. Ejemplo: ["12", "25", "48"], donde cada valor de la lista corresponde con el "id" del usuario que solicita consultar por el "flag". Entonces si un usuario con "id" igual a "12" solicita evaluar dicha "flag" el sistema va a retornar verdadero, para cualquier valor fuera de ese listado retorna falso.

Una vez registrado el "flag" el sistema genera un *token* (con encoding base64) único de acceso que el cliente debe utilizar para invocar la evaluación de la misma.

RF4. Listado de "flags"

Actor: Usuario administrador y usuario de organización

Se debe disponibilizar un listado de "flags", permitiendo filtrar por tipo, nombre, estado y fecha en que estuvo activa o no.

RF5. Baja de "flag"

Actor: Usuario administrador

Se deberá permitir dar de baja un "Flag". En caso de baja, toda consulta por ese Flag deberá responder cómo activas. No se debe eliminar el "flag", ya que debe seguir apareciendo en los reportes.

RF6. Reporte de uso

Actor: Usuario administrador y usuario de organización

Se debe disponibilizar la opción de realizar consultas sobre un "flag". Este reporte deberá mostrar los siguientes datos:

- Cantidad de veces que ha sido invocado un flag
- Tasa de respuesta positiva / negativa
- Tiempo promedio de respuesta

RF7. Evaluación de "flag"

Actor: Sistema externo

Se deberá disponibilizar un endpoint REST que permita consultar si un usuario de una aplicación tiene habilitado un "Flag". A su vez se debe mantener la información necesaria para elaboración de los reportes. Este endpoint debe responder siempre por debajo de los **100ms** para cargas de hasta **1200 req/m.**

Requerimientos no funcionales

RNF1. Performance

El tiempo de respuesta promedio para todas las operaciones públicas del sistema en condición de funcionamiento normal deberá mantenerse por debajo de los **100 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.

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 una guía de estilos de Ruby a determinar y chequeados 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. Evaluación de “flags” en formato JSON

Con el fin de evaluar una “flag” dado un usuario, se deberá brindar una API *RESTful* mínima que permita consultar por una “flag” particular dado un usuario y retornar el resultado así como también un reporte de uso de la misma, todo en formato JSON.

RNF8. 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 funcionales automatizadas para el RF 1 a nivel de sus controladores.

RNF9. 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 RNFs 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 **18 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 Flags.
- Dirección de los repositorios privados en Github.
- Cualquier conjunto de credenciales adicionales que sean necesarias para el uso de Flags.