Trabajo Práctico

Fecha de Entrega: 26 de octubre de 2021

Introducción

AlGlobo.com es un nuevo sitio de venta de pasajes online. Gracias al fin de la pandemia ha empezado a crecer fuertemente y necesitan escalar su proceso de reservas.

Para ello desean reemplazar una implementación monolítica actual con un microservicio en Rust que se encarge especificamente de este proceso.

Objetivo

Deberán implementar una aplicación en Rust que modele el sistema de reservas de AlGobo.com. Los pedidos de reserva se leerán desde un archivo y el webservice de cada aerolínea se simulará con sleeps aleatorios, y su resultado utilizando también variables aleatorias (random).

Requerimientos

El proceso de reserva se compone de la siguiente manera:

- El pedido de reserva ingresa al sistema e indica aeropuerto de origen, de destino, aerolínea preferida y si la reserva es por paquete o solo vuelo.
- El pedido se debe enviar a la aerolínea correspondiente. Para ello se comunica con un webservice especifico de la misma. Por motivos de rate limiting, cada webservice no puede realizar mas de N pedidos concurrentes (es un límite configurable). Tener en cuenta que la cantidad de aerolíneas con las que se comunica el sistema debe poder crecer con un esfuerzo mínimo.
- La aerolínea puede aceptar el pedido o recharzarlo. Si falla el sistema debe esperar X segundos para reintentar (el tiempo debe ser configurable).
- En el caso de que la reserva sea por paquete, el pedido se debe enviar también al webservice de hoteles. El resultado final de la reserva entonces necesitará que ambos pedidos (hotel y aerolinea) se completen. Por suerte las reservas de hoteles nunca se rechazan.
- El sistema debe mantener estadísticas de las rutas más solicitadas, que luego se utilizan para realizar ofertas. Para ello debe mantener un conteo de la cantidad de

reservas realizadas de cada ruta y poder informar periódicamente las 10 más solicitadas.
- Además el sistema mantendrá estadísticas operacionales. Para ello debe calcular el tiempo medio que toma una reserva desde que ingresa el pedido hasta que es finalmente aceptada.
- Se debe escribir un archivo de log con las operaciones que se realizan y sus resultados.

Parte A

Implementar la aplicación utilizando las herramientas de concurrencia de la biblioteca standard de Rust vistas en clase: Mutex, RwLock, Semáforos (del crate std-semaphore), Channels, Barriers y Condvars.

Parte B

Implementar la aplicación basada en el modelo de Actores, utilizando el framework Actix.

Puntos extra

Utilizar Tokio HTTP y/o Actix web para recibir requests reales y Apache AB para generarlos en lugar de leerlos de un archivos.

Requerimientos no funcionales

Los siguientes son los requerimientos no funcionales para la resolución de los ejercicios:

- El proyecto deberá ser desarrollado en lenguaje Rust, usando las herramientas de la biblioteca estándar.
- No se permite utilizar **crates** externos, salvo los explícitamente mencionados.
- El código fuente debe compilarse en la última versión stable del compilador y no se permite utilizar bloques unsafe.
- El código deberá funcionar en ambiente Unix / Linux.
- El programa deberá ejecutarse en la línea de comandos.
- La compilación no debe arrojar **warnings** del compilador, ni del linter **clippy**.
- Las funciones y los tipos de datos (**struct**) deben estar documentadas siguiendo el estándar de **cargo doc**.
- El código debe formatearse utilizando **cargo fmt**.
- Cada tipo de dato implementado debe ser colocado en una unidad de compilación (archivo fuente) independiente.

Entrega

La resolución del presente proyecto es en grupos de tres integrantes.

La entrega del proyecto comprende lo siguiente:

- Informe, se deberá presentar en forma digital (PDF) enviado por correo electrónico a la dirección: pdeymon@fi.uba.ar
- El código fuente de la aplicación, que se entregará únicamente por e-mail. El código fuente debe estar estructurado en un proyecto de cargo, y se debe omitir el directorio target/ en la entrega. El informe a entregar debe contener los siguientes items:
    - Una explicación del diseño y de las decisiones tomadas para la implementación de la solución.
    - Detalle de resolución de la lista de tareas anterior.
    - Diagrama que refleje los threads, el flujo de comunicación entre ellos y los datos que intercambian.
    - Diagramas de entidades realizados (structs y demás).

Criterios de evaluación

Presentación, principios teóricos y defensa de bugs potenciales

Los alumnos presentarán el código de su solución en vivo en una reunión sincrónica, con foco en el uso de las diferentes herramientas de concurrencia. Deberán poder explicar desde los conceptos teóricos vistos en clase cómo se comportará potencialmente su solución ante problemas de concurrencia (por ejemplo ausencia de deadlocks).

En caso de que la solución no se comportara de forma esperada, deberán poder explicar las causas y sus posibles rectificaciones.

Casos de prueba en vivo

Durante la presentación se someterá a la aplicación a diferentes casos de prueba que validen la correcta aplicación de las herramientas de concurrencia.

Informe

El informe debe estar estructurado profesionalmente y debe poder dar cuenta de las decisiones tomadas para implementar la solución.

Se debe detallar en un diagrama, las entidades desarrolladas, las herramientas de concurrencia empleadas. Así como también los threads y formas de comunicación entre ellos. Se debe poder entender qué mensajes datos entre ellos y de qué forma.

Organización del código

El código debe organizarse respetando los criterios de buen diseño y en particular aprovechando las herramientas recomendadas por Rust (i.e. no utilizar unsafe)

Tests automatizados

La presencia de tests automatizados que prueben diferentes casos, en especial sobre el uso de las herramientas de concurrencia es un plus.

Presentación en término

El trabajo deberá entregarse para la fecha estipulada. La presentación fuera de término sin coordinación con antelación con el profesor influye negativamente en la nota final.

Participación individual

Si bien el trabajo es grupal, la nota es individual y la participación del alumno durante la presentación influye en su nota final.