Juan Fernando Castañeda Gutiérrez
201623311

Explicación A2

El proyecto entero se encuentra contenido en una única clase, App. La forma en que App maneja concurrencia es simulándola internamente. **App es al mismo tiempo la aplicación que ve el usuario y el simulador de concurrencia.**

Los métodos a los que accede el usuario son:

- App.start
- App.register_app
- App.get_registered_apps
- App.unregister_app
- App.update_and_get_requests
- App.choose_service
- App.update_and_get_current_service

Los métodos de la simulación son:

- App.app_behaviour
- App.send_requests_until_unregistered
- App.register_request
- App.generate_service_specs

La cantidad de parámetros y el significado de cada uno está en el archivo documentado. Sin embargo, se va a exponer un **caso común de simulación**:

1. El usuario empieza la app con App.start.
2. El usuario registra la primera app con App.register_app(“Uber”, 10).
3. El usuario revisa que se agregó la aplicación con App.get_registered_apps.
4. La simulación empieza a simular a través de App.app_behaviour(1). En este caso es 1 porque es la primera app agregada.
5. El usuario puede ver las requests que le están llegando a su bandeja con App.update_and_get_requests.
6. El usuario tiene que ser rápido a la hora de elegir una request para tomar, porque estas desaparecen. La manera en que las elige es con App.choose_service(id de la request que vio en el paso anterior).
7. El usuario revisa que el servicio se agregó con App.update_and_get_current_service.
8. El usuario desregistra la aplicación que agregó con App.unregister_app(1). En este caso es 1 porque es la primera app agregada.
9. El usuario puede revisar que ya no le lleguen más requests de esa app y que ya no aparezca registrada con los comandos anteriormente mencionados. Desregistrar la app elimina también todas las requests asociadas, menos la actual, en caso de que esté haciendo.

Hay que tener en cuenta que este proceso puede ser hecho con cuantas apps simuladas al tiempo se quiera, porque es concurrente. En cuanto a la **concurrencia**, el programa utiliza mutex en todas las actualizaciones que pueden tener los agentes. No se identifica ningún deadlock porque no fue necesario anidar mutexses; no hacerlo no afectaba el comportamiento del programa gracias a el diseño de los métodos de retornar, que al mismo tiempo actualizaban.