Tesztelés

Tesztelés menete

Az általam definiált tesztesetekhez tartoznak szkriptek, amelyek a futásuk során bizonyos időközönként kérnek adatot a szolgáltatótól, és ezek eredményét egy külön fájlba mentik.
A tesztelés automatizálásának kivitelezése nehezebbnek bizonyult, mint gondoltam, és idő és szakértelem híján nem sikerült úgy megoldanom, ahogy azt szerettem volna.
A tesztelés az alábbiak szerint működik:

- A tesztelés megkezdése előtt a szerveralkalmazás adatbázisát fel kell tölteni adatokkal, a tesztek az1,2,3,4 hotel ID-kat használják, illetve 6-os és 8-as azonosítót használnak az érvénytelen kliensek. Az IRFHotels alkalmazást el lehet indítani ClearData parancssori bemenettel, ekkor törlődik az adatbázis tartalma.
- A szkriptek csak a klienseket indítják el, tehát a szerveralkalmazások indításáról nekünk kell gondoskodni. Lehetőleg minden teszt futtatása előtt indítsunk új kiszolgálót, mert több teszt elvárt eredménye azzal számol, hogy a kliensek és a kiszolgáló egyidőben indultak.
- A szkripteknek bemenetként meg kell adni a kliensalkalmazáshoz tartozó exe fájlt abszolút elérési úttal együtt. (Valamiért nem sikerült, amikor relatív eléréssel próbáltam elindítani.)
- A szkriptekben nem foglalkoztam hibakezeléssel.

A feltöltött megoldásomban a tesztek egy korábbi futtatásának kimenetei is megtalálhatók.

Tesztesetek és elvárt kimeneteik

Test1

Ebben a tesztben egyetlen kliensalkalmazás fog futni nagyjából egy percig. A szolgáltatótól a futás elején és végén kérek le adatokat.
A teszt során az AvgAliveMessagesPerMin számláló működése könnyen ellenőrizhető, hiszen 1 kliensnek, 1 perc alatt nagyjából 20 kérést kellene küldenie.

Test2

Ebben a tesztben 4 hotelt regisztrálok, és mind a négyhez indítok is klienst. Az alkalmazásokat azonban röviddel a futás kezdete után, 10 mp-es időközönként, egyesével leállítom. A szolgáltatót minden leállítás után lekérdezem.
A teszt során a ActiveHotels, RateOfActiveHotels és a TimeOuts számlálók működését ellenőrizhetjük könnyen.
Az ActiveHotels számlálónak 4-ről 0-ra kell csökkenie, ezzel párhuzamosan a RateOfActiveHotels-nek pedig 1-ről negyedenként 0-ra.
A TimeOutsnak pedig 0-ról 4-re kell nőnie fokozatosan.

Test3

Ebben a tesztben 4 kliens fut, ezek közül 2 érvényes, azaz regisztrált hotelhez tartozik, 2 pedig érvénytelen. A kliensek egyszerre, nagyjából 1 percig fognak futni. A szolgáltatótól a teszt elején és végén fogok adatokat lekérdezni.

Ezzel a teszttel a RateOfInvalidAliveMessages metrika helyessége ellenőrizhető könnyen, hiszen kétszer annyi érvényes kliensünk fut, így a számláló értékének a futás végére olyan 0.5 körüli értéket kellene mutatnia.

Test4

Ebben a tesztben 2 érvényes kliens fut, egyszerre, nagyjából 2 percig. Azonban az egyik hotelt 6, a másikat pedig 12 másodpercenként leállítom 6 másodpercre (hogy biztosan timeoutoljanak). A futás elején és végén kérdezem le a szolgáltatót.
Ebben a tesztben az AvgActiveTime mérhető jól. Mivel a kliensek fele-fele arányban futnak 6 illetve 12 másodpercig az átlagos aktivitási időnek nagyjából 9 mp-nek kellene lennie.

Test5

Ebben a tesztben 3 érvényes kliens fut, egyszerre, nagyjából 1 percig. Mindehárom hotelben 4 szoba található, kezdetben mind üres. Az egyik hotelt az 1 perc alatt teljesen feltöltöm, a másikat szintén feltöltöm, de itt a foglalások felét vissza is mondom, a harmadik hotelhez nem nyúlok.
Itt tehát az AvgRoomReservationsPerMin és a RateOfBookedActiveHotels számlálók lesznek az érdekesek.
Az első 30 mp-ben a két hotelt feltöltöm, ekkor tehát azt kell látnunk, hogy az AvgRoomReservationsPerMin számláló 16 körüli értéket mutat, hiszen fél perc alatt 8 szobát foglaltunk, a RateOfBookedActiveHotels pedig 2/3.
A második 30 mp-ben a második hotel foglalásainak felét visszamondom. Mire letelik az egy perc az AvgRoomReservationsPerMin értéke 4 körüli, a RateOfBookedActiveHotels pedig 1/3.