25. Sinkronizacija procesa razmjenom obavijesti

Izvještaj

Bruno Blažeka

Martin Pisačić

## Opis rješenja

Kao temu projekta za laboratorij profila odabrali smo programiranje raspodijeljenog sustava za nadzor zračenja koji je dio sustava zaštite nuklearne elektrane. Kroz taj primjer smo odlučili primjeniti naučeno gradivo i ostvariti sinkronizaciju procesa razmjenom obavijesti.

U našem sustavu postoje senzori, korisnici i posrednik. Posrednik koordinira obavijesti od senzora prema korisnicima i ima ulogu posrednika u procesu sinkronizacije.

Senzori nam služe kao izvor mjerenja i svoja mjerenja generiraju čitanjem iz datoteke. Mjerenja se generiraju svakih 5 sekundi i zapisana su u mjernoj jedinici: µSv/h.

Svakom mjerenju se pridružuje status u ovisnosti o pročitanoj vrijednosti:

|  |  |  |
| --- | --- | --- |
| **Normal** | **Warning** | **Critical** |
| Manje od 1 | Između 1 i 7 | Veće od 7 |

Posrednik obavijesti usmjerava prema pretplatnicima ovisno o njihovoj razini (pretplatnik se prilikom pretplate opredjeljuje za određenu razinu), a pretplatnici u tom sustavu prestavljaju izlaz prema korisniku sustava za obavještavanje.

Tipovi izlaza mogu biti: spremanje u bazu (sve razine), stavljanje zapisa na web stranicu (ako je razina „Warning“ ili „Critical“) ili kontaktiranje nadležne službe – postavlja uzbunu (razina je „Critical“).

## Tehničke specifikacije

Korištene tehnologije: .NET Framework 4.5.1, C#, RabbitMQ

Korišteno razvojno okruženje: Visual Studio 2017

## Struktura rješenja

Programsko rješenje se sastoji od nekoliko projekata:

|  |  |
| --- | --- |
| Base | Projekt s pomoćnim klasama koje koriste drugi projekti iz aplikacije tijekom svog rada (npr. kod operacija čitanja i pisanja, rad s razinama opasnosti) |
| Broker | Projekt posrednika, sastoji se od klase koja raspršuje obavijesti na pretplatnike |
| GeigerCounterSystem | Projekt za simuliranje senzora koji čita iz fileova i pridjeljuje razine opasnosti. |
| MessageCommunication | Projekt s klasama koje koriste za komunikaciju između sudionika sustava. Koristi se za stvaranje različitih komunikacijskih kanala za različite tipove obavijesti i za jednostavniji prijenos podataka po njima. |
| Starter | Projekt koji započinje generiranje podataka u senzorima |
| User | Projekt koji opisuje ponašanje korisničkog procesa u kojem se pretplaćuje na obavijesti razine koja ga zanima |

Poveznica na Github repozitorij na kojem je pohranjeno rješenje:

<https://github.com/bblazeka/TemporalProcessSynchronization>

## Pokretanje rješenja

Napisati naredbu u PowerShell konzolu:

rassusStart {path do direktorija u kojem je clone repozitorija}

Nakon izvršavanja naredbe, projekti će se buildati i pokrenuti.

Pokreću se procesi:

* Starter – generira mjerenja
* Broker – ima ulogu posrednika
* User – nekoliko procesa s različitim pretplatama na obavijesti

## Opis rada rješenja

Tijekom rada sustava, akcije pokreće proces „Starter“ generiranjem obavijesti. Proces „Starter“ generirane podatke obrađuje i šalje rezultate mjerenja i proračunate razine na posrednika. Proces ima i ispis u konzolu gdje evidentira sve što je poslano na posrednika.

Proces „Broker“ zapisuje sva primljena mjerenja u konzolu te šalje pretplaćenim procesima obavijesti na uspostavljene kanale.

Prilikom pokretanja procesa „User“ se zadaju pretplate na koje se pretplaćuje pojedini korisnik, ovisno o svojoj ulozi (npr. Ako ga zanimaju samo pretplate „Critical“). Zbog toga, procesi „User“ primaju samo neke obavijesti (one koje odgovaraju njihovoj funkciji) te ih ispisuju u konzolu. Ispisivanje u konzolu prestavlja njihov izlaz na web i slično.