Konzistentan sistem

Softver nadzorno-upravljačkih sistema

Tim: MAP

Petrić Mladen SV68-2022

Cetković Aleksa SV77-2022

Dragičević Petar SV12-2022

Fakultet tehničkih nauka, Novi Sad

Uvod

Tema projekta je razvoj distribuiranog sistema za očitavanje i sinhronizaciju temperatura korišćenjem WCF (Windows Communication Foundation) servisa. Sistem obuhvata tri senzora koji nezavisno mere temperaturu i upisuju merenja u svoje baze podataka, kao i centralnu komponentu — koordinatora, koji obezbeđuje konzistentnost podataka između senzora. Cilj projekta je implementirati konsenzusni mehanizam koji omogućava pouzdano očitavanje podataka čak i u slučaju da pojedini senzori daju neispravne ili zastarele vrednosti.

Opis sistema

Sistem se sastoji od tri glavne komponente:

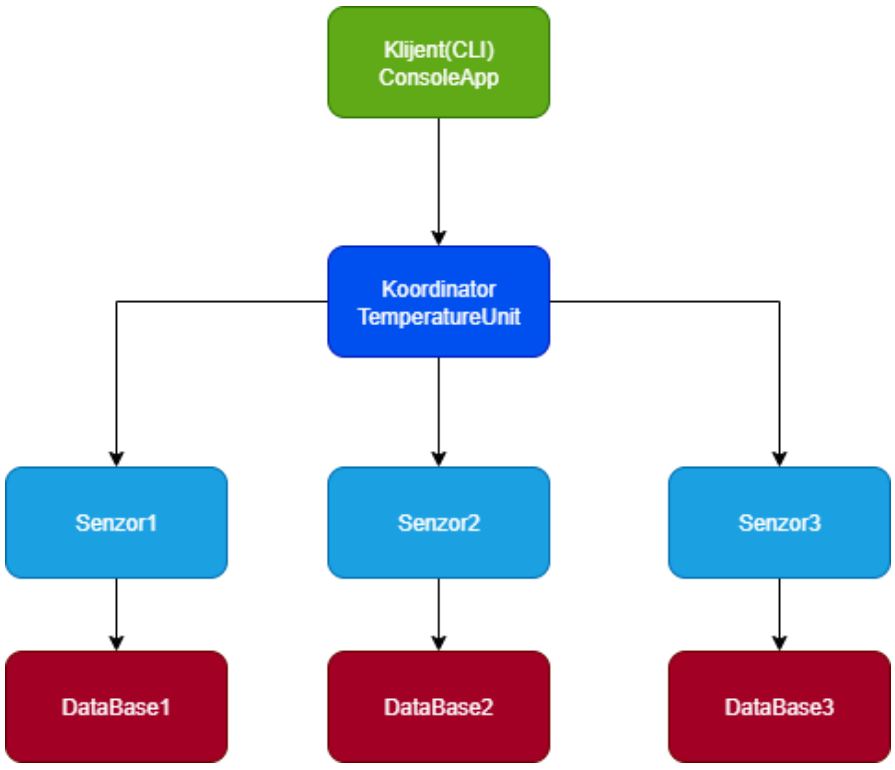
- **TemperatureSensorService** – predstavlja simulirani fizički senzor. Svaki senzor se pokreće kao poseban WCF servis i poseduje sopstvenu bazu podataka (PostgreSQL). Na svakih 1–10 sekundi nasumično generiše novo merenje temperature i upisuje ga u svoju bazu.

- **TemperatureUnit (Koordinator)** – komponenta koja preko WCF-a komunicira sa svim senzorima. Njegova uloga je da: prikupi sva merenja, proveri kvorum (da li se najmanje 2 senzora nalaze unutar ±5°C od prosečne vrednosti), vrati rezultat ako postoji konsenzus, ili pokrene proces poravnanja (sync) ako do konsenzusa nije došlo.

- **Klijentska aplikacija (ConsoleApp)** – predstavlja korisničku aplikaciju koja koristi TemperatureUnit da bi prikazala očitane temperature, validirala podatke i prikazala status sistema.

Arhitektura sistema

Svaki senzor komunicira preko HTTP binding-a (BasicHttpBinding) i izlaže WCF endpoint na različitom portu. Koordinator koristi ChannelFactory da dinamički otkrije i komunicira sa svakim senzorom.



TemperatureSensorService

Ova komponenta implementira interfejs ITemperatureSensor sa metodama:

- **double ReadTemperature();** - vraća poslednje merenje iz baze.
- **void SyncTemperature(double temperature);** - poziva kada dođe do poravnanja senzora — upisuje novu sinhronizovanu vrednost u bazu.

Senzor automatski generiše nova merenja pomoću Timer klase u intervalima od 1 do 10 sekundi. Podaci se čuvaju u PostgreSQL bazi pomoću Npgsql biblioteke.

TemperatureUnit (Koordinator)

Klasa TemperatureUnit predstavlja centralnu tačku sistema i zadužena je za:

- čitanje podataka sa svih senzora paralelno
- formiranje liste očitanja i izračunavanje proseka
- utvrđivanje kvoruma — da li najmanje 2 senzora daju slične vrednosti (u okviru ±5°C)
- pokretanje procesa Sync() ako kvorum nije postignut

Mehanizam zaključavanja (ReaderWriterLockSlim) obezbeđuje da se ne može čitati dok traje poravnanje, čime se postiže konzistentnost sistema. Takođe, TemperatureUnit ima automatski tajmer koji na svakih 60 sekundi samostalno pokreće poravnanje svih senzora, nezavisno od aktivnosti klijenta.

ConsoleApp (Klijentska aplikacija)

Glavna aplikacija (Program.cs) predstavlja centralnu tačku za pokretanje i testiranje kompletnog sistema. Ona obavlja ulogu klijenta, ali istovremeno u okviru istog procesa pokreće sve WCF servise neophodne za rad sistema. Prilikom pokretanja, aplikacija izvršava sledeće korake:

- Kreira i pokreće tri instance senzora (TemperatureSensorService) na različitim portovima: 8000, 8001 i 8002. Svaki senzor funkcioniše kao nezavisni WCF servis sa sopstvenom bazom podataka.
- Kreira i pokreće TemperatureUnit servis (koordinator) na portu 8004, koji komunicira sa senzorima preko WCF-a.

Nakon što su svi servisi aktivni, aplikacija kreira WCF klijenta pomoću klase ChannelFactory<ITemperatureUnit> i poziva metodu ReadTemperature() iz koordinatora. Dobijeni rezultat se ispisuje u konzoli — prikazuje se trenutna temperatura ili poruka o neusaglašenosti senzora ukoliko konsenzus nije postignut.

CAP teorema

CAP teorema definiše kompromis između:

- **C (Consistency)** – svi čvorovi imaju iste podatke
- **A (Availability)** – sistem odgovara na svaki zahtev
- **P (Partition tolerance)** – sistem nastavlja rad i pri greškama mreže

Naš sistem je CP – daje prednost konzistentnosti nad dostupnošću. Tokom poravnanja (Sync()), čitanja moraju da sačekaju, što obezbeđuje da se u svim bazama na kraju nalaze identične vrednosti.

Kvorum i poravnanje

Kvorum od 2/3 znači da sistem zahteva najmanje dve slične vrednosti kako bi smatrao merenje pouzdanim. Ukoliko je razlika između prosečne vrednosti svih senzora i bilo kog očitanja veća od ±5°C, sistem pokreće poravnanje (synchronization). Poravnanje izračunava srednju vrednost poslednjih merenja i postavlja je kao novu temperaturu u svim senzorima.

Zaključak

Projekat Konzistentan sistem prikazuje implementaciju distribuiranog sistema sa osnovnim mehanizmima kvoruma i održavanja konzistentnosti. Upotrebom WCF servisa, PostgreSQL baza i synchronization lock mehanizama, postignut je sistem koji ispunjava zahteve konzistentnosti čak i u prisustvu neusaglašenih senzora. Moguća poboljšanja uključuju dodavanje grafičkog korisničkog interfejsa, naprednije politike odlučivanja pri poravnanju i podršku za više senzora.