



**FER3**

**Diplomski studij**

## **Raspodijeljeni sustavi**

Prva laboratorijska vježba —  
Prikupljanje i obrada senzorskih  
podataka

**Ak. god. 2023./2024.**

# 1 Uvod

Cilj ove laboratorijske vježbe je izgraditi centralizirani raspodijeljeni sustav temeljen na modelu klijent-poslužitelj za senziranje parametara okoliša kako je definirano u odjeljku 2. Često postavljena pitanja nalaze se u odjeljku 5. Tijekom izrade programskog rješenja naučit ćete koristiti tehnologije gRPC<sup>1</sup> i web-usluge temeljene na prijenosu prikaza stanja resursa (REST). Ova se vježba sastoji od tri dijela:

- proučavanja primjera programskog kôda s predavanja (gRPC i web-usluge),
- oblikovanje i implementacija poslužitelja raspodijeljenog sustava koji održava informacije o senzorima i
- oblikovanje i implementacija klijenta raspodijeljenog sustava koji predstavlja senzor za parametre okoliša.

## 1.1 Predaja

Studenti samostalno rješavaju programski zadatak i dužni su putem sustava Moodle predati arhivu koja se sastoji od:

- izvornog kôda poslužitelja i
- izvornog kôda klijenta.

**Napomena:** Komponente poslužitelja i klijenta moraju biti implementirane kao dva odvojena projekta u odabranom razvojnom okruženju (npr. Eclipse, NetBeans, Visual Studio, IntelliJ Idea).

Arhiva se mora predati do roka koji će biti objavljen na stranici predmeta. Studenti koji predaju vježbu nakon navedenog roka dobit će 0 bodova iz vježbe.

Uz predaju digitalne arhive, organizirat će se **usmena prezentacija** predanog rješenja kada će studenti morati objasniti implementaciju i pokazati svoje razumijevanje primijenjenih komunikacijskih tehnologija, a kako bi se rješenje moglo ocijeniti. Termin i provedba usmenog ispita, tj. kolokviranje vježbe, bit će objavljeni na stranici predmeta. Kolokviranje je nužno kako bi predano rješenje bilo ocijenjeno.

Sve komponente sustava mogu se implementirati u bilo kojem programskom jeziku. Imajte na umu da su primjeri prikazani na predavanjima implementirani u programskom jeziku Java. Arhivu koja sadrži izvorni kôd treba nazvati **ime\_prezime\_jmbag**. Arhiva treba sadržavati 2 direktorija koja sadrže datoteke izvornog kôda (jedan za klijenta i jedan za poslužitelja). Dopušteno je slanje zip arhive koja sadrži izvoz projekta ako je implementiran u IDE-u (Eclipse, NetBeans itd.).

Konzultacije se održavaju **utorkom od 10:00 do 11:00**, uz prethodnu **najavu** na MS Teamsu (kanal Laboratorijske vježbe) ili e-mailom.

# 2 Arhitektura raspodijeljenog sustava

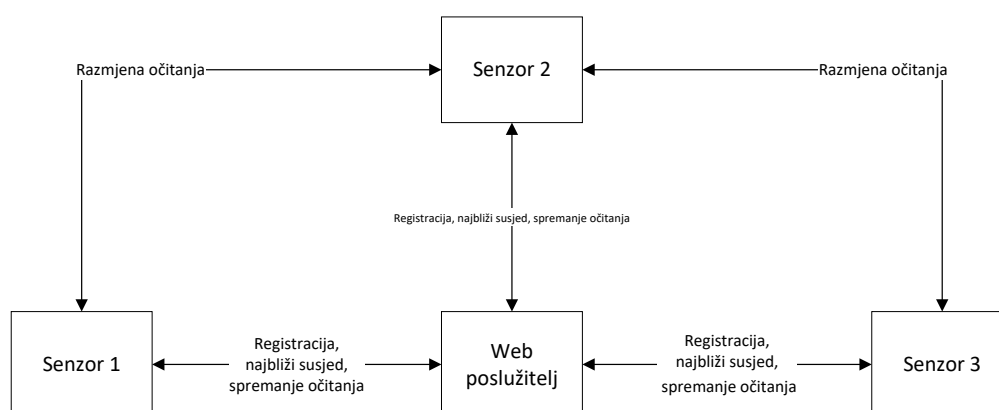
**Kratak pregled funkcionalnosti sustava:** Raspodijeljeni sustav koristi se za praćenje parametara okoliša u većem geografskom području u koje su postavljeni senzori (svaki senzor ima svoju geo-lokaciju) pomoću poslužitelja implementiranog kao web-usluga i skupine senzora. Arhitektura raspodijeljenog sustava prikazana je na slici 1.

Zadaća senzora je generiranje senzorskih očitavanja za parametre zraka, npr. temperature, tlaka, vlage, itd. te razmjena i usporedba vlastitih očitavanja s očitanjima geografski najbližeg senzora. U primjeni senzori često mogu generirati pogrešna očitavanja, pa ćemo stoga korištenjem vlastitih očitavanja

<sup>1</sup><https://grpc.io/>

i očitavanja s najbližeg susjeda kreirati tzv. kalibrirana očitavanja. Programski se klijent raspodijeljenog sustava (u daljnjem tekstu **senzor**) sastoji se od **klijenta web-usluge**, **gRPC klijenta** i **gRPC poslužitelja**. Funkcionalnost senzora i komunikacija između senzora detaljno su objašnjeni u poglavlju 3.

Zadaća poslužitelja je upravljanje podacima o dostupnim registriranim sensorima, njihovom geografskom položaju i kalibriranim očitanjima. Svi senzori komuniciraju s poslužiteljem koji nudi web-usluge temeljene na prijenosu prikaza stanja resursa (REST) (u daljnjem tekstu **poslužitelj**) koristeći **JSON** format. Funkcionalnost poslužitelja i komunikacija između senzora i poslužitelja detaljno su objašnjene u poglavlju 4.



Slika 1: Arhitektura raspodijeljenog sustava

## 3 Funkcionalnost senzora

Ključne funkcije senzora uključuju mogućnost inicijalizacije i registracije na poslužitelju, generiranje vlastitih očitavanja, razmjenu vlastitih očitavanja sa svojim geografski najbližim susjedom putem gRPC veze, mogućnost kalibriranja vlastitih i razmijenjenih očitavanja te mogućnost pohranjivanja kalibriranih očitavanja na poslužitelju. Prilikom implementacije senzora, možete koristiti priloženi kôd na stranicama predmeta kao primjer za razvoj gRPC klijenta i gRPC poslužitelja. Za senzore se može koristiti bilo koji HTTP klijent (npr. Retrofit<sup>2</sup>). Postupak generiranja očitavanja, dohvaćanje očitavanja od susjeda, kalibriranje očitavanja i slanje kalibriranih očitavanja na poslužitelj senzor neprestano ponavlja sve dok korisnik ne zaustavi proces.

### 3.1 Inicijalizacija i registracija

Nakon pokretanja, senzor nasumično odabire geografsko mjesto u rasponu između [15.87, 16] za zemljopisnu dužinu (eng. *longitude*) i [45.75, 45.85] za zemljopisnu širinu (eng. *latitude*). Mjesto se ne mijenja tijekom vijeka trajanja senzora. Senzor se nakon toga mora registrirati na poslužitelju. Poruke razmijenjene tijekom registracije prikazane su u poglavlju 4.1.

### 3.2 Generiranje senzorskih očitavanja

Budući da pravi senzori nisu spojeni na klijente, njihova se funkcija emulira pripremljenim očitanjima iz ulazne datoteke **readings.csv**, gdje očitavanja (redci) imaju više vrijednosti (stupaca) koje predstavljaju parametre. Za generiranje očitavanja iz ulazne datoteke upotrijebite jednadžbu 1 za dobivanje broja retka koji ćete koristiti za očitavanje temperature, barometarskog tlaka, relativne vlažnosti i koncentracije plinova CO i NO2 ili SO2 iz ulazne datoteke:

<sup>2</sup><https://square.github.io/retrofit/>

$$red \leftarrow (brojAktivnihSekundi \% 100) + 1 \quad (1)$$

Varijabla *brojAktivnihSekundi* je vrijeme rada senzora (vrijeme se mjeri od trenutka pokretanja klijenta i izražava se u sekundama).

### 3.3 Razmjena očitavanja

Nakon što senzor generira vlastito očitavanje, mora zatražiti očitavanje od svog geografski najbližeg susjeda putem gRPC veze. Za pronalaženje geografski najbližeg susjeda senzor koristi poslužitelja koji pohranjuje sve podatke o senzorima sustava. Izgled poruka koje senzor razmjenjuje s poslužiteljem tijekom ovog procesa definiran je u poglavlju 4.1. Nakon pronalaska najbližeg susjeda, senzor započinje proces razmjene očitavanja sa susjedom tako što mu šalje zahtjev za dostavu očitavanja (zahtjev definirajte proizvoljno), a susjedni senzor odgovara slanjem vlastitih očitavanja (vodite računa o parametrima očitavanja). Pretpostavite da se najbliži susjed senzora ne mijenja nakon pokretanja senzora. Nije potrebno tražiti novog najbližeg susjeda ako postojeći najbliži susjed više nije dostupan. Međutim, senzor mora ostati u funkciji odnosno predvidjeti mogućnost da najbliži susjed više nije dostupan. Ako senzor nema najbližeg susjeda, ne kalibrira svoja očitavanja, već vlastita očitavanja pohranjuje na poslužitelju.

**Napomena:** Sensori moraju biti u mogućnosti održavati više istovremenih komunikacijskih kanala, tj. komunicirati istodobno s više senzora kojima su potrebna njihova očitavanja (tada su u ulozi gRPC poslužitelja) i otvoriti kanal za komunikaciju s maksimalno jednim susjednim senzorom (tada su u ulozi gRPC klijenta). Senzor treba ispisati odgovarajuću poruku na svom sučelju kada se gRPC veza prekine i obraditi iznimku. Tijekom procesa očitavanja novih vrijednosti potrebno je ispisati relevantne informacije na sučelju senzora (odnosno koristiti **logging**).

### 3.4 Kalibriranje očitavanja

Senzor koristi vlastita očitavanja i očitavanja dobivena od susjeda za **kalibraciju**, odnosno senzor potvrđuje kvalitetu vlastitih očitavanja uspoređujući ih s očitanjima susjednog senzora. Kalibracija podataka vrši se izračunavanjem prosječne vrijednosti između dva očitavanja (prvo je vlastito očitavanje, a drugo je očitavanje sa susjednog senzora). Tijekom kalibracije izračunava se prosjek za iste parametre očitavanja (tj. temperatura, tlak zraka, relativna vlažnost i koncentracije plinova CO, NO<sub>2</sub> i SO<sub>2</sub>).

**Napomena:** Čvorovi senzora korišteni za izradu ulazne datoteke sadrže samo dva senzora plina (CO, NO<sub>2</sub> ili SO<sub>2</sub>), pa biste trebali biti svjesni mogućeg **odsustva nekih vrijednosti plina prilikom čitanja ulazne datoteke**. Ako neke vrijednosti nedostaju ili ako je izmjerena vrijednost postavljena na 0 (tj. došlo je do pogreške u očitavanju senzora), tada upotrijebite samo vlastitu vrijednost za prosjek i obrnuto. Ako i vlastita vrijednost i vrijednost susjeda nedostaju, isključite tu vrijednost iz prosjeka (na primjer, postavite je na null).

### 3.5 Spremanje očitavanja

Nakon što su podaci kalibrirani, senzor šalje kalibrirana očitavanja poslužitelju pomoću odgovarajuće metode web-usluge. Točne poruke razmijenjene tijekom spremanja očitavanja prikazane su u poglavlju 4.3.

## 4 Funkcionalnost poslužitelja i komunikacija sa senzorima

Senzori i poslužitelj komuniciraju putem REST sučelja i pomoću JSON formata. Iako se bilo koji jezik može koristiti za implementaciju poslužitelja, predavanja pokrivaju uporabu Spring Boota<sup>3</sup>. Predložak

<sup>3</sup><https://start.spring.io/>

za poslužitelja dostupan je u materijalima kolegija. Predavanja o ovim tehnologijama dostupna su na **poveznici**.

### 4.1 Registracija

Senzor se registrira slanjem registracijskog zahtjeva s relevantnim podacima (njegova geo-lokacija i adresa gRPC poslužitelja) poslužitelju. Primjer podataka za registraciju prikazuje JSON 1. Poslužitelj dodjeljuje identifikator novostvorenom senzoru. Poslužitelj treba vratiti **201 Created** i URL na stvoreni resurs u zaglavlju **HTTP Location**. Ako se senzor nije uspješno registrirao (npr. server nije dostupan), treba prekinuti rad.

**Napomena:** obratite pažnju na vrstu HTTP zahtjeva, kao i URL na koji se zahtjev šalje.

JSON 1: Primjer zahtjeva za registraciju u JSON formatu

```
1 {
2   "latitude": 45.75,
3   "longitude": 15.87,
4   "ip": "127.0.0.1",
5   "port": 8080
6 }
```

### 4.2 Najbliži susjed

Nakon registracije na poslužitelju, senzor može od poslužitelja zatražiti svog geografski najbližeg susjeda. Udaljenost između dva senzora izračunava se pomoću Haversineove formule koja je definirana pseudokôdom 1 (6731 km je radijus Zemlje). Primjer odgovora na zahtjev geografski najbližeg susjeda prikazuje JSON 2.

Pseudokod 1: Haversineova formula

```
1 R ← 6371;
2 dlon ← lon2 − lon1;
3 dlat ← lat2 − lat1;
4 a ← (sin(dlat/2))2 + cos(lat1) * cos(lat2) * (sin(dlon/2))2;
5 c ← 2 * atan2(sqrt(a), sqrt(1 − a));
6 d ← R * c;
```

JSON 2: Primjer odgovora na zahtjev za geografski najbližeg susjeda u JSON formatu

```
1 {
2   "latitude": 45.76,
3   "longitude": 15.88,
4   "ip": "127.0.0.1",
5   "port": 8081
6 }
```

### 4.3 Spremanje očitavanja pojedinog senzora

Senzor može pohraniti svoja kalibrirana očitavanja na poslužitelju slanjem odgovarajućeg zahtjeva. Poslužitelj treba vratiti **201 Created** i URL na kreirani resurs u zaglavlju HTTP Location headera. Primjer podataka u zahtjevu za spremanje očitavanja prikazuje JSON 3. U slučaju da ne postoji senzor sa identifikatorom za kojeg se očitavanje sprema, poslužitelj treba vratiti odgovor **204 No Content**.

**Napomena:** senzor generira više očitavanja, ali jedno očitavanje pripada jednom senzoru (odnosa među entitetima je 1 :  $N$ ). Obratite pažnju na vrstu HTTP zahtjeva, kao i URL na koji se zahtjev šalje.

JSON 3: Primjer zahtjeva za spremanje očitavanja u JSON formatu

```
1 {  
2   "temperature": 32.0,  
3   "pressure": 1109.0,  
4   "humidity": 40.0,  
5   "co": 547.0,  
6   "so2": 18.0  
7 }
```

#### 4.4 Popis senzora

Poslužitelj treba izložiti popis registriranih senzora u JSON formatu. Primjer odgovora na zahtjev za popis senzora prikazuje JSON 4.

JSON 4: Primjer odgovora na zahtjev za popis registriranih senzora u JSON formatu

```
1 [  
2   {  
3     "id": 1,  
4     "latitude": 45.75,  
5     "longitude": 15.87,  
6     "ip": "127.0.0.1",  
7     "port": 8080  
8   },  
9   {  
10    "id": 2,  
11    "latitude": 45.76,  
12    "longitude": 15.88,  
13    "ip": "127.0.0.1",  
14    "port": 8081  
15  }  
16 ]
```

#### 4.5 Popis očitavanja pojedinog senzora

Poslužitelj treba izložiti popis očitavanja koja pripadaju određenom senzoru. Primjer odgovora na zahtjev za popis očitavanja prikazuje JSON 5. U slučaju da ne postoji senzor sa identifikatorom za kojeg se traži popis očitavanja, poslužitelj treba vratiti odgovor **204 No Content**.

## JSON 5: Primjer odgovora na zahtjev za popis očitavanja JSON formatu

```
1  [  
2    {  
3      "id": 4,  
4      "temperature": 27.0,  
5      "pressure": 749.5,  
6      "humidity": 46.5,  
7      "co": 174.5,  
8      "no2": 128.0,  
9    },  
10   {  
11     "id": 6,  
12     "temperature": 28.0,  
13     "pressure": 749.5,  
14     "humidity": 46.5,  
15     "co": 174.5,  
16     "no2": 125.0,  
17   }  
18 ]
```

## 5 Često postavljena pitanja

### 5.1 Smijem li koristiti jezik i/ili razvojno okruženje po svom odabru?

Smijete, uz opasku da će za 3. laboratorijsku vježbu biti obavezno koristiti Spring Boot odnosno Javu.

### 5.2 Koji jezik i/ili razvojno okruženje nam preporučujete?

Preporučujemo korištenje:

1. Java 17
2. Gradle 7.5.1
3. org.springframework.boot 2.7.4
4. io.spring.dependency-management 1.0.14.RELEASE

### 5.3 Kako pokrenuti gRPC primjer?

1. gradle runExampleServer (pokreće SimpleUnaryRPCServer)
2. gradle runExampleClient (pokreće SimpleUnaryRPCClient)

### 5.4 Postoje li upute o tome kako napraviti POST podataka?

Predavanja o Springu i REST-u možete naći:

1. U sklopu on-line predavanja<sup>4</sup>
2. Razni oblici REST klijenata za Javu<sup>5</sup>

<sup>4</sup>[https://www.youtube.com/watch?v=Wtjx-75w5CY&list=PLy0T81VDh93YLJEEe5AxydDLXxUPrPs\\_B](https://www.youtube.com/watch?v=Wtjx-75w5CY&list=PLy0T81VDh93YLJEEe5AxydDLXxUPrPs_B)

<sup>5</sup>[https://www.youtube.com/watch?v=3zTxHA2Fn0o&list=PLy0T81VDh93YLJEEe5AxydDLXxUPrPs\\_B&index=10](https://www.youtube.com/watch?v=3zTxHA2Fn0o&list=PLy0T81VDh93YLJEEe5AxydDLXxUPrPs_B&index=10)

## 5.5 Pitanja studenata prijašnjih godina

**Q1: Kada od poslužitelja tražim najbližeg susjeda, poslužitelj šalje odgovor koji se sastoji od sljedećih vrijednosti: latitude, longitude, ip, port. Problem se pojavljuje u sljedećem koraku: susjedni senzor trebao bi poslati svoje najaktualnije očitavanje (pretpostavljam da bi se to najaktualnije očitavanje ponovo trebalo dohvatiti s poslužitelja), a ja nemam identifikator susjednog senzora - ne mogu napraviti zahtjev na poslužitelj tipa GET /readings/sensor/id (dohvati očitavanja susjednog senzora).**

A1: Senzor svoje najaktualnije očitavanje čita iz .csv datoteke, ne s poslužitelja. Što se tiče očitavanja najbližeg susjeda, senzor A će od svojeg najbližeg susjeda (recimo senzora B) dobiti očitavanje putem GRPC veze, a ne Spring poslužitelja. Na slici 1 je dan okvirni tok informacija u sustavu. Komunikacija je opisana u poglavlju 3.3 u rečenici:

"Nakon što senzor generira vlastito očitavanje, mora zatražiti očitavanje od svog geografski najbližeg susjeda putem gRPC veze."