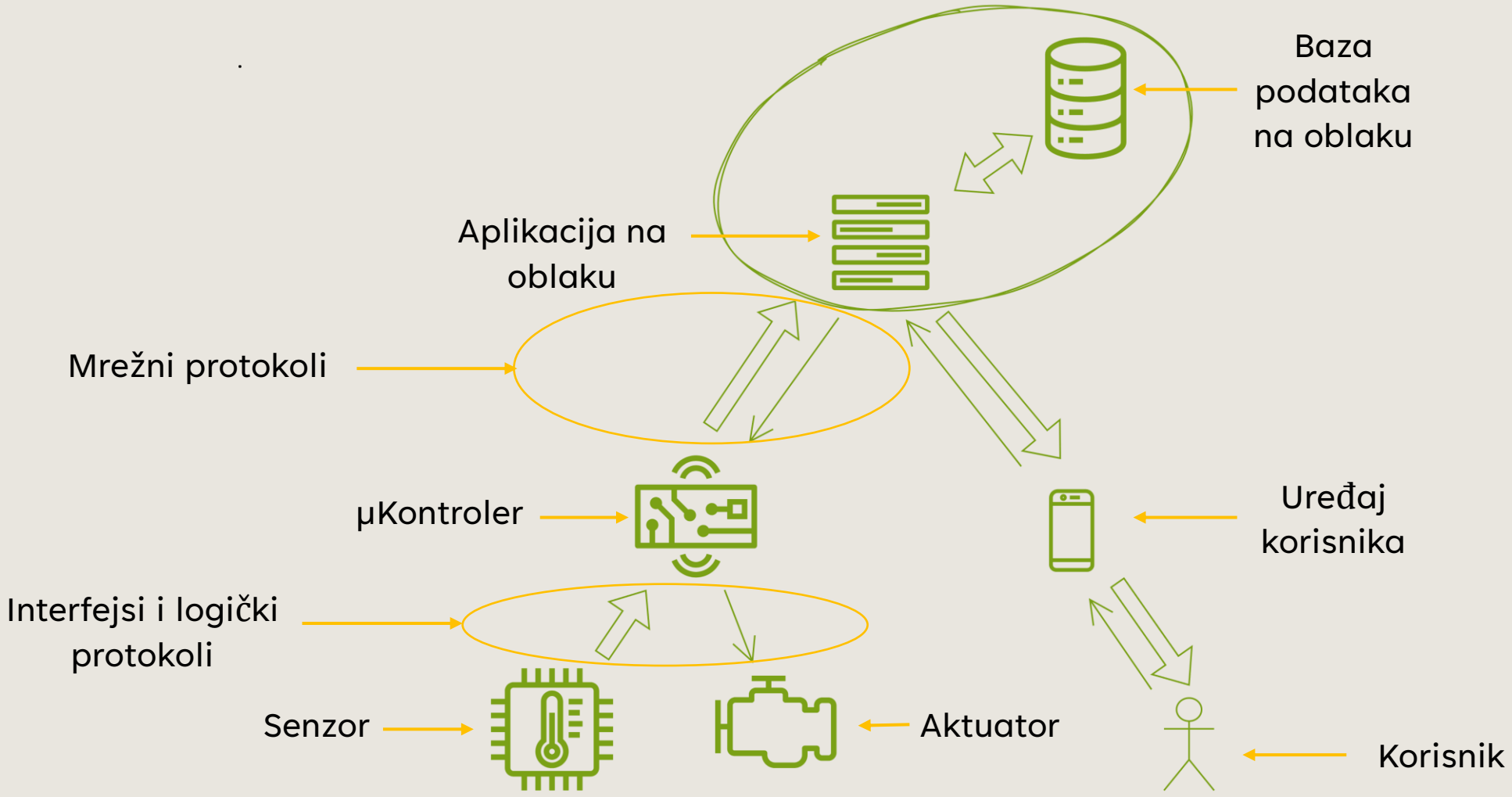
INTERNET PAMETINIH UREĐAJA

prof. dr Dejan S. Aleksić

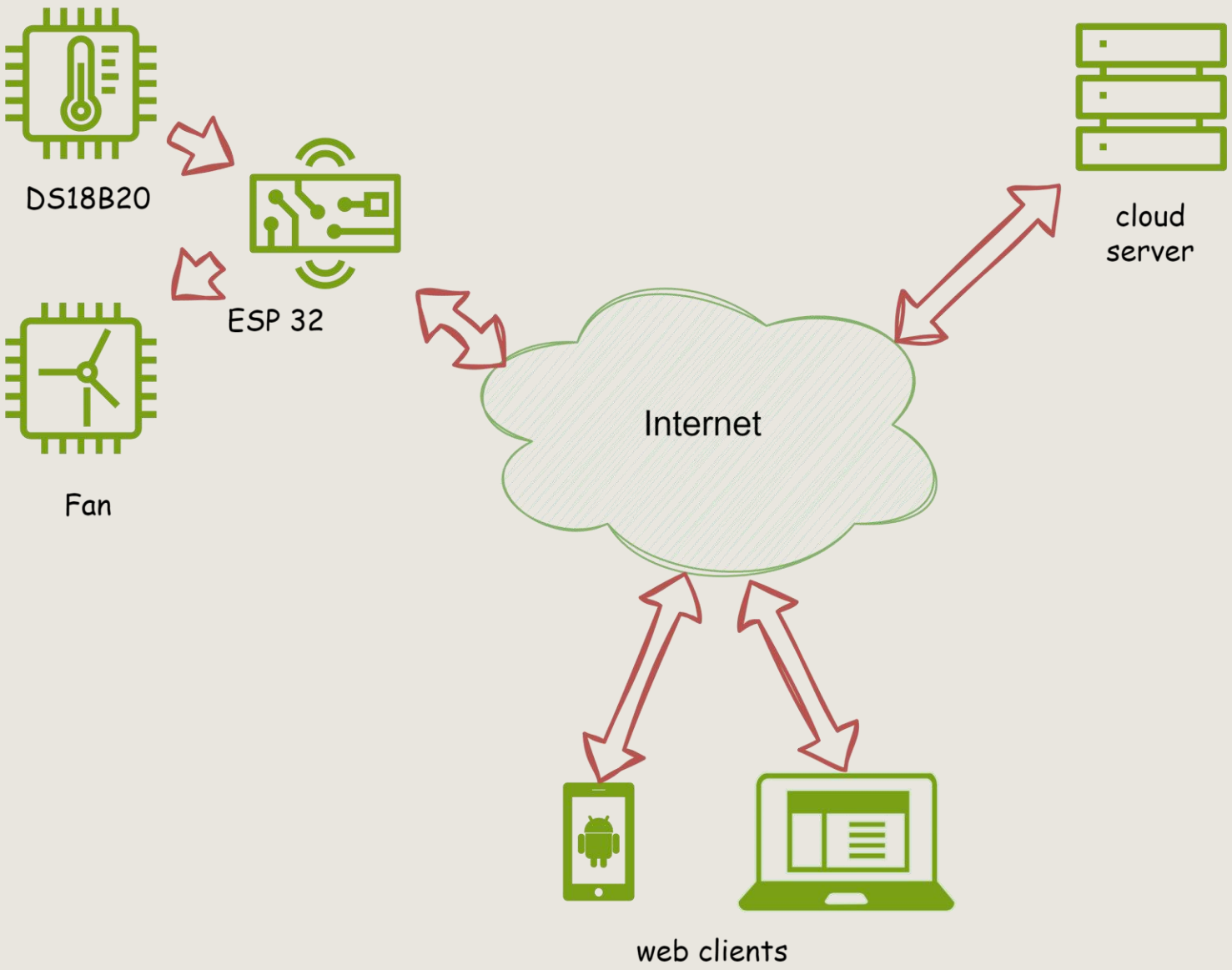
Prirodno-matematički fakultet, Niš

07. APLIKACIJA

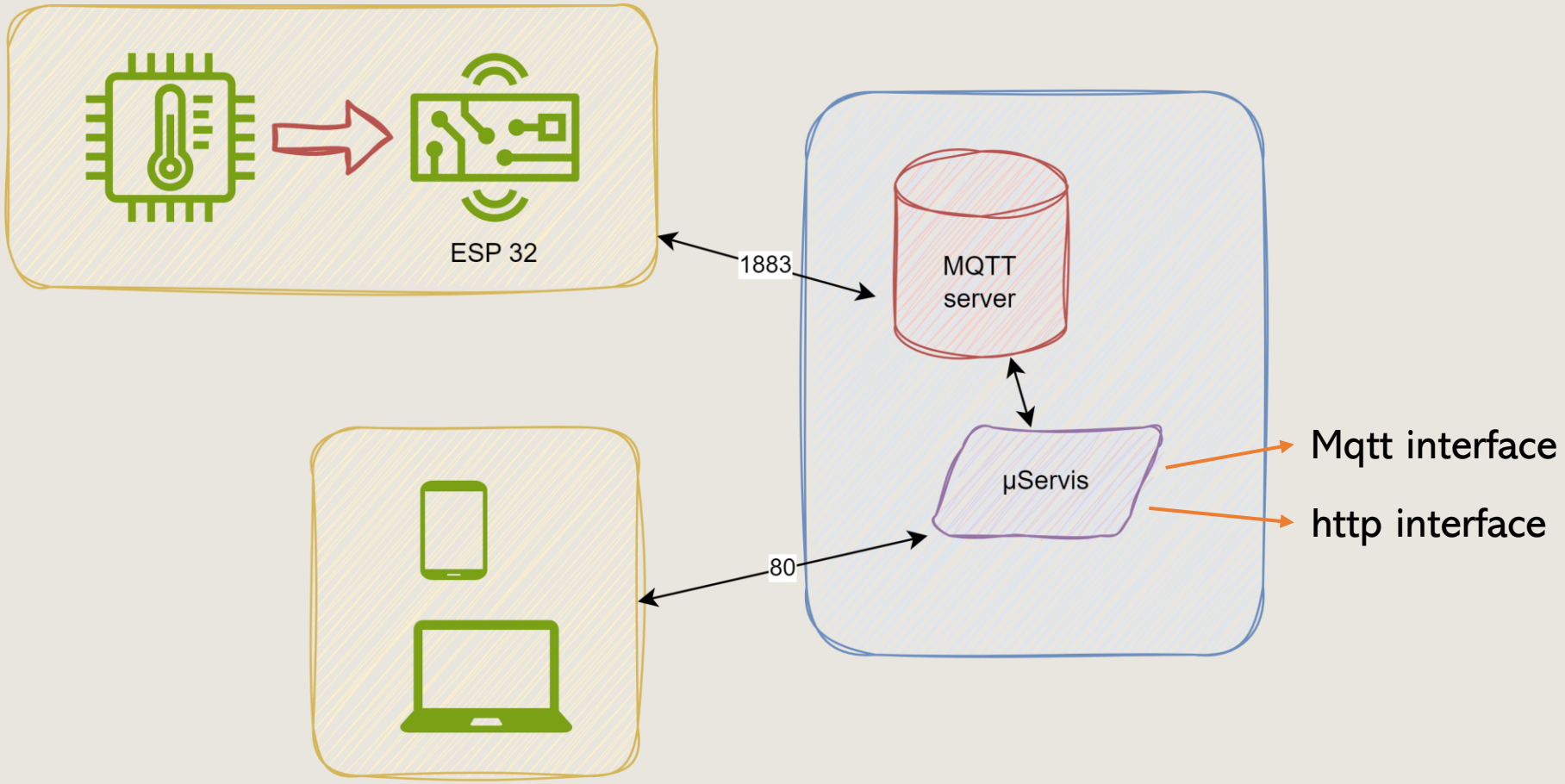
OPŠTI KONCEPT APLIKACIJE



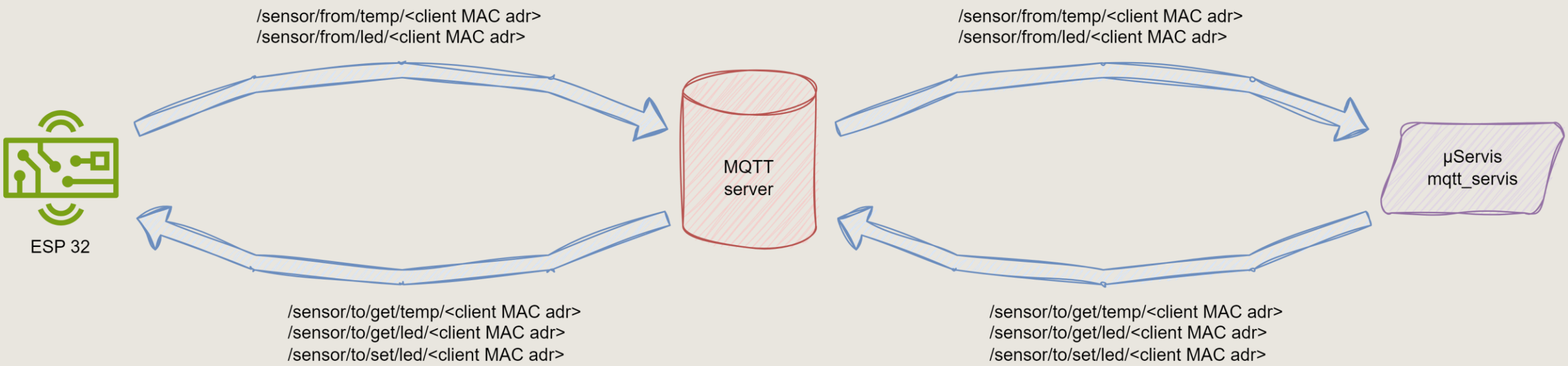
OPŠTI KONCEPT APLIKACIJE



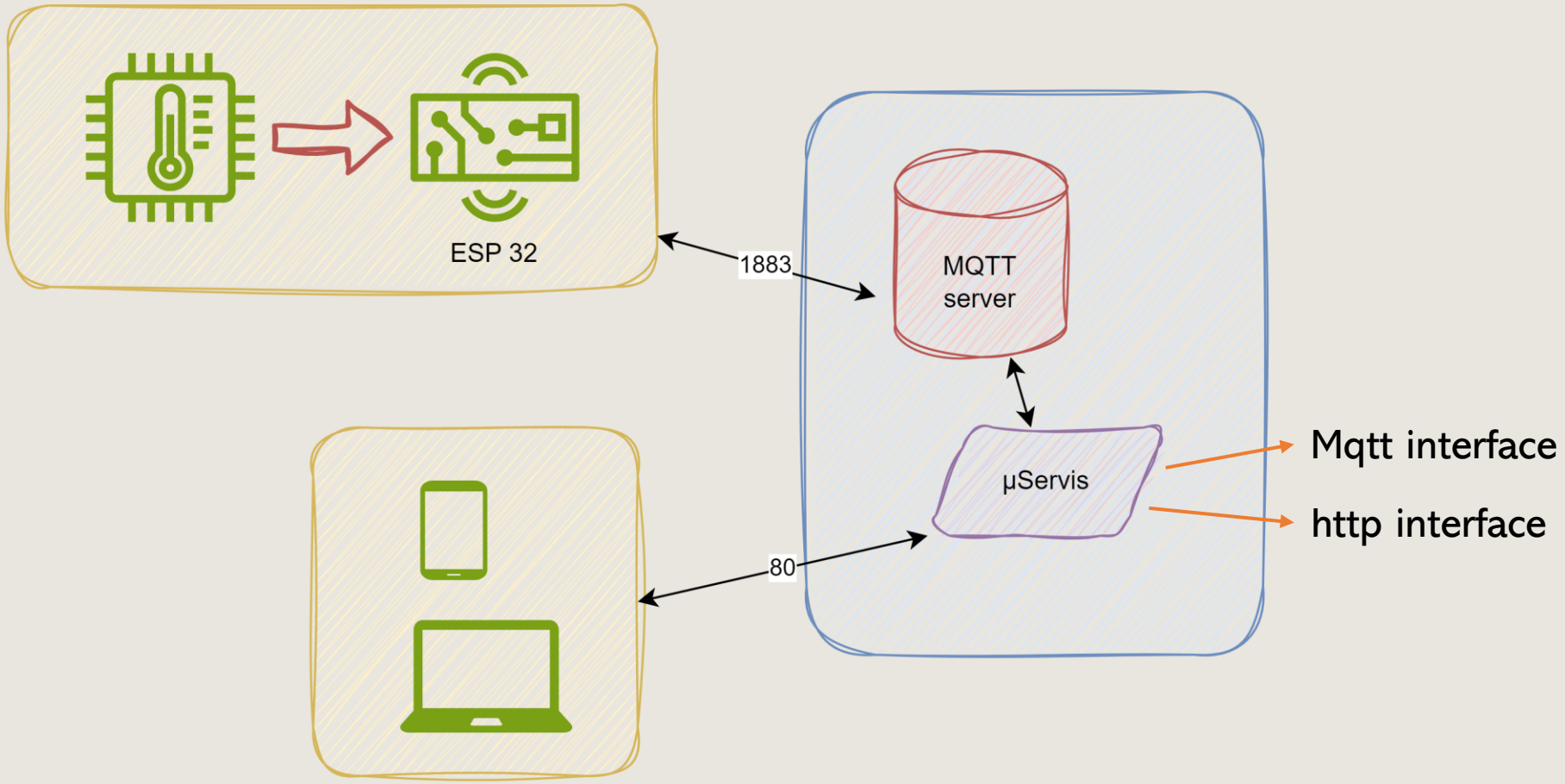
REŠENJE 1.


ESP 32
1883
MQTT
server
µServis
Mqtt interface
http interface
80

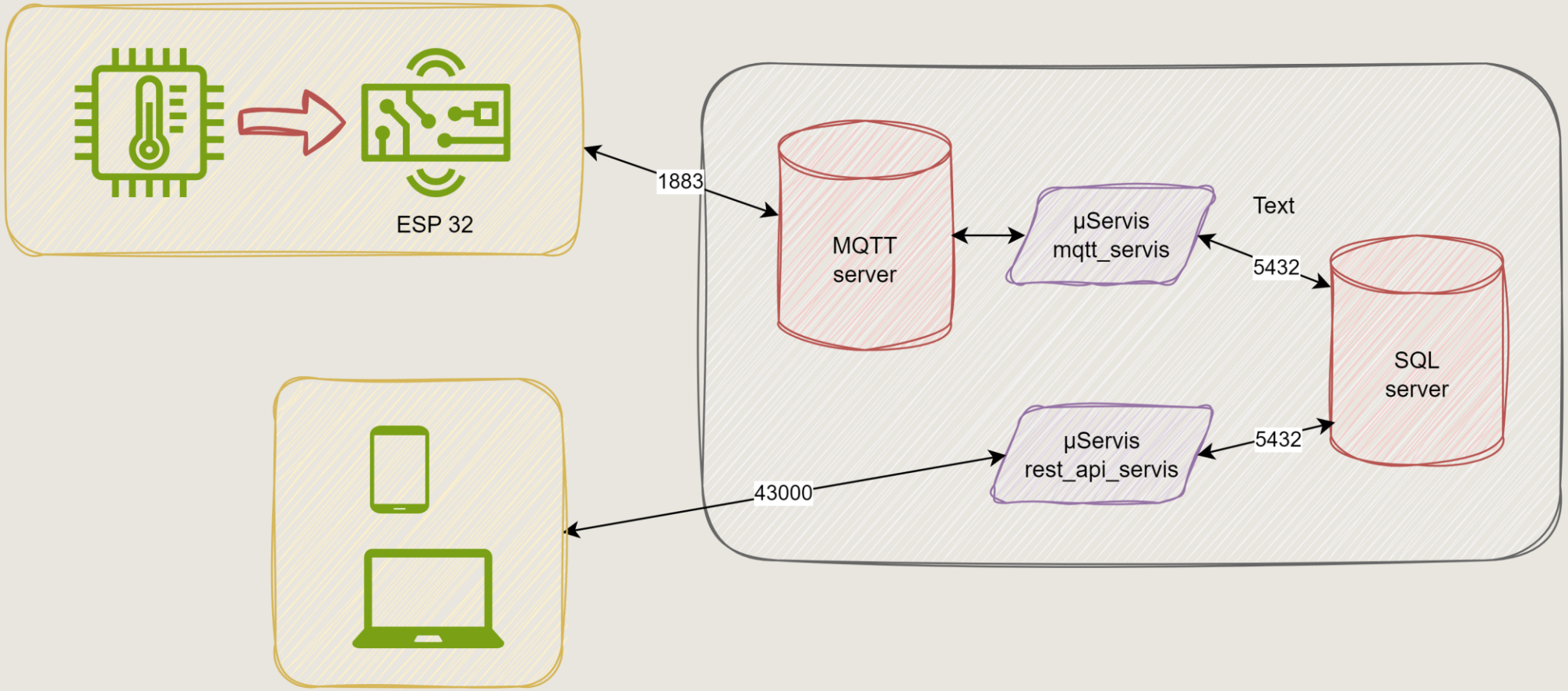
OPŠTI KONCEPT APLIKACIJE



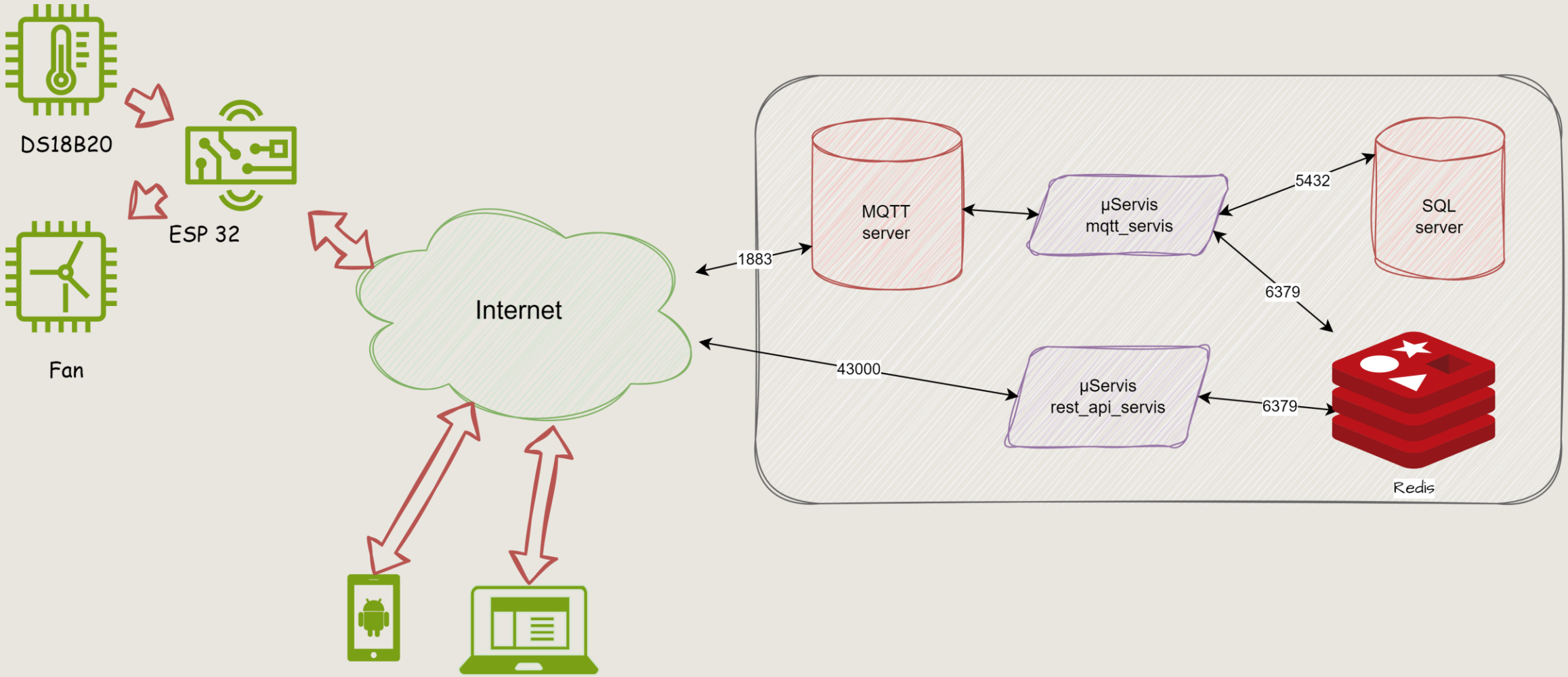
REŠENJE 1.


ESP 32
1883
MQTT
server
µServis
Mqtt interface
http interface
80

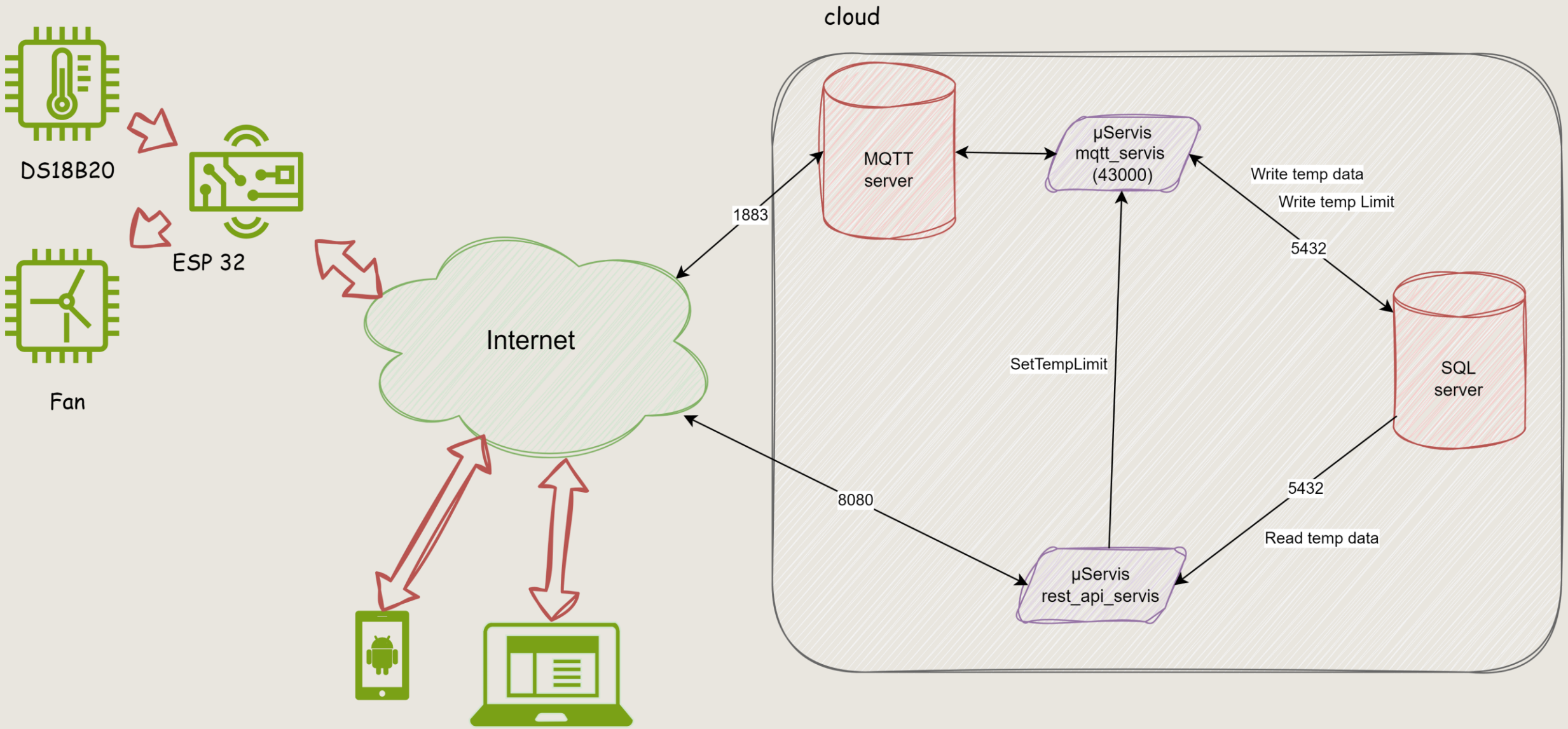
REŠENJE 2.


ESP 32
1883
MQTT
server
µServis
mqtt_servis
Text
5432
SQL
server
µServis
rest_api_servis
5432
43000

REŠENJE 3.

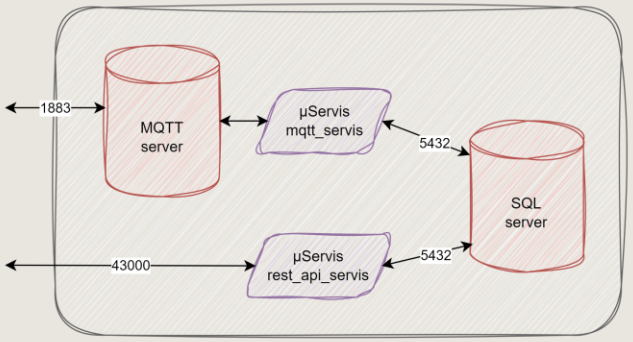

DS18B20
ESP 32
Fan
Internet
1883
MQTT
server
µServis
mqtt_servis
5432
SQL
server
6379
43000
µServis
rest_api_servis
6379
Redis

REŠENJE 3.

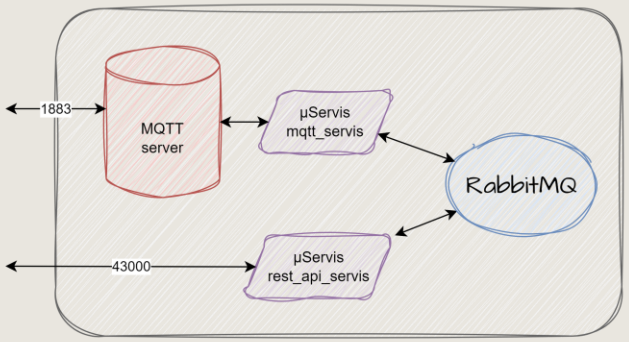

cloud
DS18B20
ESP 32
Fan
Internet
1883
MQTT server
µServis mqtt_servis (43000)
Write temp data
Write temp Limit
5432
SQL server
SetTempLimit
8080
5432
Read temp data
µServis rest_api_servis

RAZMENA PODATAKA IZMEĐU MIKRO SERVISA

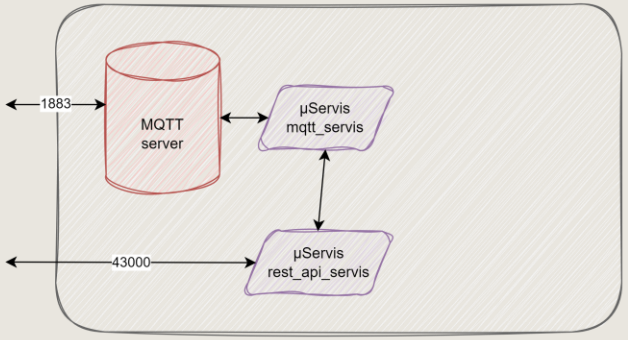
db Alert



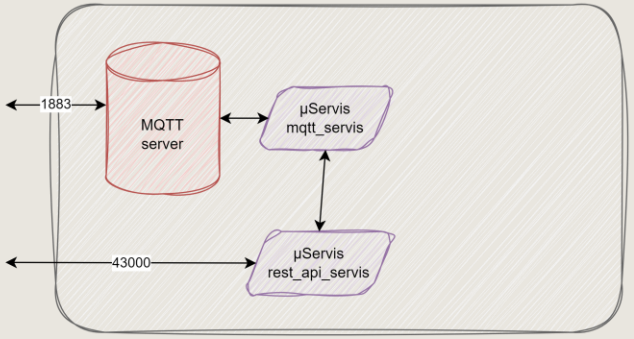
Komunikacija preko redova poruka



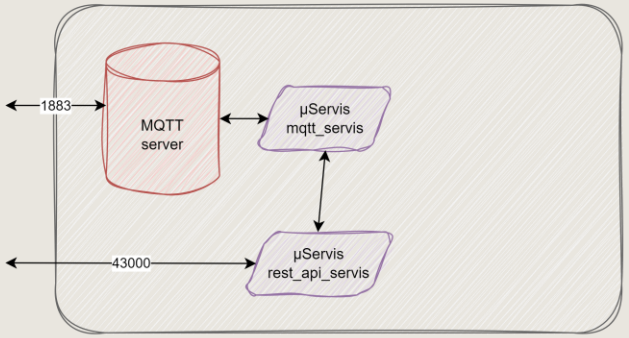
Rest API pozivi



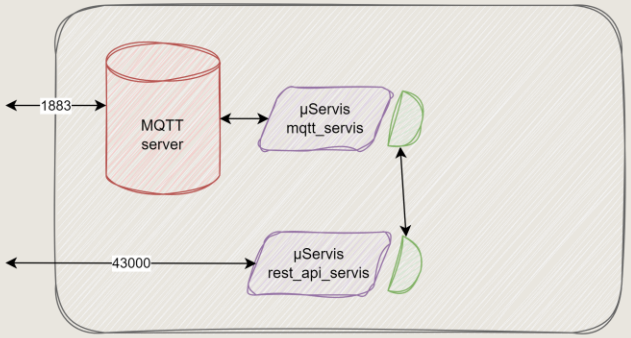
gRPC



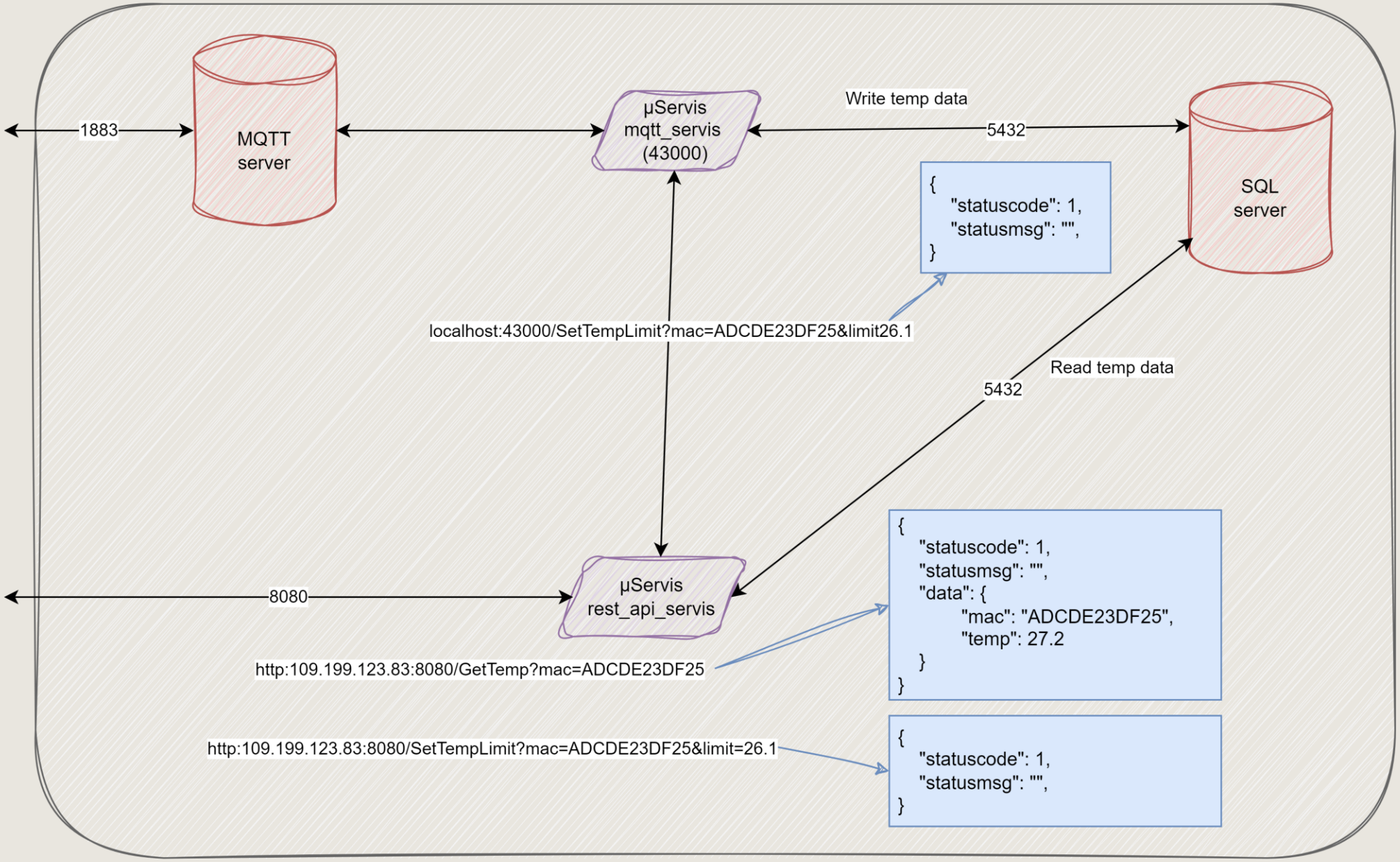
GraphQL



Service meshes



RAZMENA PODATAKA IZMEĐU MIKRO SERVISA



PARTICIONIRANJE TABELE.

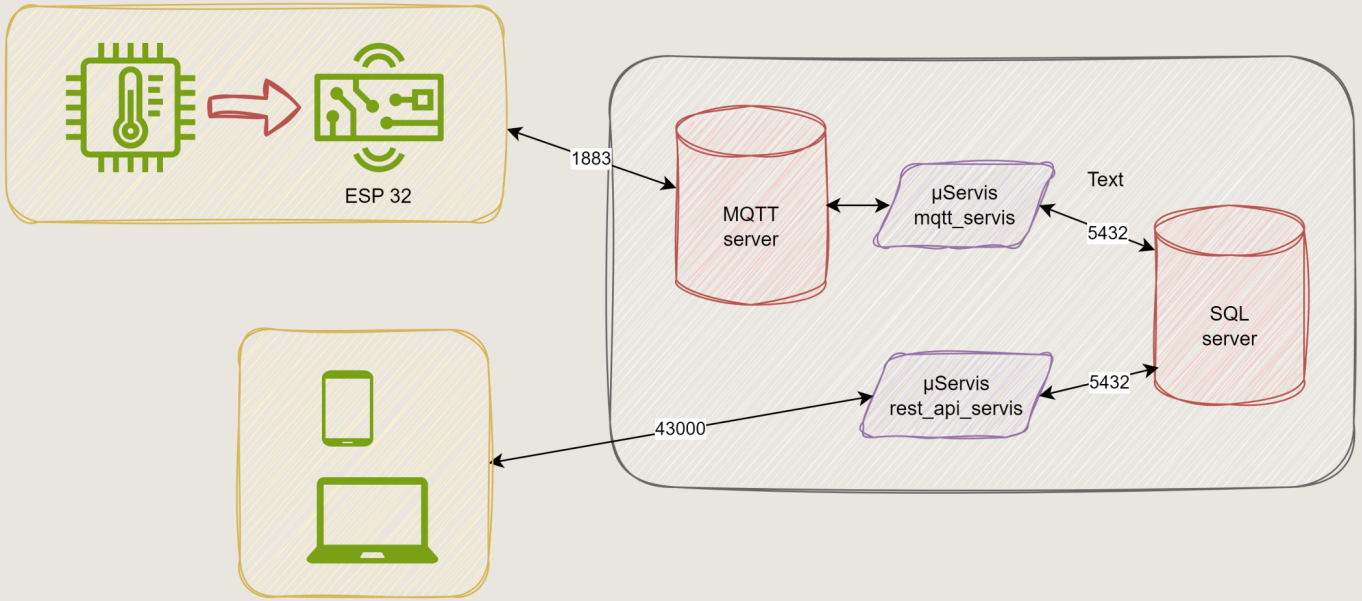


Table Partitioning

- Particioniranje tabele je tehnika dizajna baze podataka koja se koristi za podelu velike tabele na manje delove kojima se lakše upravlja, nazvane particije.
- Svaka particija je u suštini zasebna tabela koja čuva podskup originalnih podataka.
- Ova tehnika može značajno da poboljša performanse upita i upravljanje podacima za velike skupove podataka.
- Particionisanje se može izvršiti na osnovu jedne ili više kolona, kao što je kolona datuma ili opseg vrednosti.
- Na primer, možete da podelite tabelu na osnovu datuma zapisa, gde svaka particija predstavlja podatke za određeni period.
- Prilikom upita podataka, PostgreSQL može brzo eliminisati particije koje nisu relevantne za upit, što rezultira bržim izvršavanjem upita.

PARTICIONIRANJE TABELE.

Prednosti podele tabele

- **Poboljšane performanse upita:** Particioniranje omogućava bazi podataka da brzo suzi podatke na određenu particiju, smanjujući količinu podataka koje treba skenirati tokom upita. Ovo rezultira bržim vremenom izvršavanja upita, posebno za velike skupove podataka.
- **Lakše upravljanje podacima:** Sa particionisanjem tabele, možete lako da upravljate velikim skupovima podataka tako što ćete ih podeliti na manje particije kojima je lakše upravljati. Ovo može da pojednostavi zadatke kao što su arhiviranje podataka, čišćenje podataka i operacije pravljenja rezervnih kopija i vraćanja.
- **Poboljšano učitavanje i indeksiranje podataka:** Prilikom učitavanja podataka u particionisanu tabelu, proces se može paralelizirati, što dovodi do bržeg unosa podataka. Pored toga, indeksi na particionisanim tabelama mogu biti efikasniji, jer treba da pokriju samo manji podskup podataka.
- **Uštede u skladištenju:** Particionisanje vam omogućava da čuvate starije podatke ili podatke kojima se ređe pristupa na jeftinijim medijumima za skladištenje, dok čuvate podatke kojima se često pristupa na bržim uređajima za skladištenje.

PARTICIONIRANJE TABELE.

Metode particionisanja u PostgreSQL-u

PostgreSQL nudi različite metode particionisanja, uključujući:

- Particionisanje opsega
- Particionisanje liste
- Hash Partitioning

PARTICIONIRANJE TABELE.

Particionisanje opsega u PostgreSQL-u

Particionisanje opsega je tip particionisanja tabele gde se podaci dele na particije na osnovu određenog opsega vrednosti u koloni.
Ovo je posebno korisno kada se radi sa podacima vremenske serije ili bilo kojim podacima koji imaju prirodan red.
Svaka particija predstavlja poseban opseg vrednosti, a podaci koji spadaju u taj opseg se čuvaju u toj particiji.
Particionisanje opsega omogućava efikasno preuzimanje podataka unutar određenih opsega, što dovodi do poboljšanih performansi upita.

```
CREATE TABLE sales (
sale_id SERIAL,
sale_date DATE NOT NULL,
product_id INT,
quantity INT,
amount NUMERIC,
PRIMARY KEY (sale_id)
) PARTITION BY RANGE (sale_date);
```

PARTICIONIRANJE TABELE.

Kreirajte particije u PostgreSQL-u

- Napravićemo pojedinačne tabele za predstavljanje svake particije, od kojih svaka pokriva određeni raspon datuma.
- U svrhu demonstracije, napravićemo tri particije:
  - „sales_january“,
  - „sales_february“ i
  - „sales_march“.

```
CREATE TABLE sales_january PARTITION OF sales
FOR VALUES FROM ('2023-01-01') TO ('2023-02-01');

CREATE TABLE sales_february PARTITION OF sales
FOR VALUES FROM ('2023-02-01') TO ('2023-03-01');

CREATE TABLE sales_march PARTITION OF sales
FOR VALUES FROM ('2023-03-01') TO ('2023-04-01');
```

PARTICIONIRANJE TABELE.

Upis podataka u particije u PostgreSQL-u

- Sada možemo da ubacimo podatke u tabelu prodaje, a PostgreSQL će automatski usmeriti podatke na odgovarajuću particiju na osnovu sale_date:

```
INSERT INTO sales (sale_date, product_id, quantity, amount)
VALUES ('2023-01-15', 101, 5, 100.00);

INSERT INTO sales (sale_date, product_id, quantity, amount)
VALUES ('2023-02-20', 102, 10, 200.00);

INSERT INTO sales (sale_date, product_id, quantity, amount)
VALUES ('2023-03-10', 103, 8, 150.00);
```

PARTICIONIRANJE TABELE.

Čitanje podataka iz particije u PostgreSQL-u

Prilikom upita podataka, PostgreSQL će automatski pristupiti samo relevantnim particijama na osnovu klauzule WHERE.

```
-- Pribavljanje podataka za mesec januar
SELECT * FROM sales WHERE sale_date >= '2023-01-01' AND sale_date < '2023-02-01';

-- Pribavljanje podataka za mesec februar
SELECT * FROM sales WHERE sale_date >= '2023-02-01' AND sale_date < '2023-03-01';

-- Pribavljanje podataka za mesec mart
SELECT * FROM sales WHERE sale_date >= '2023-03-01' AND sale_date < '2023-04-01';
```

Ovi upiti će pristupiti samo odgovarajućim particijama, što će rezultirati poboljšanim performansama upita.