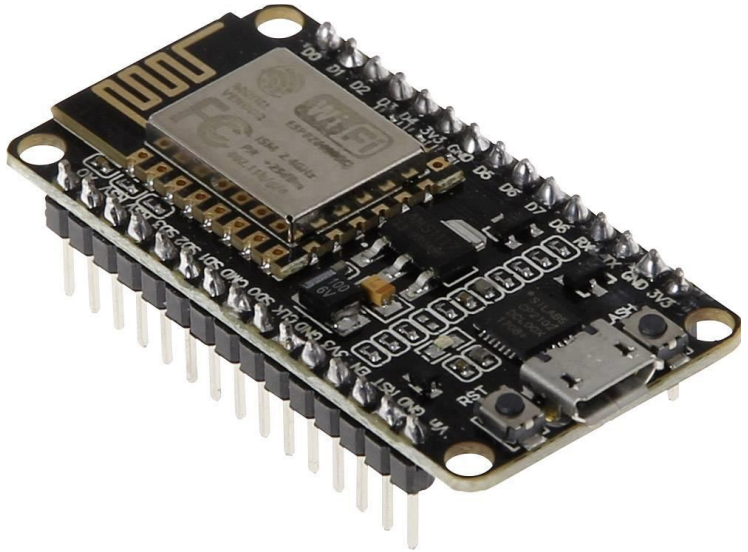
Otthoni adatgyűjtő rendszer fejlesztése

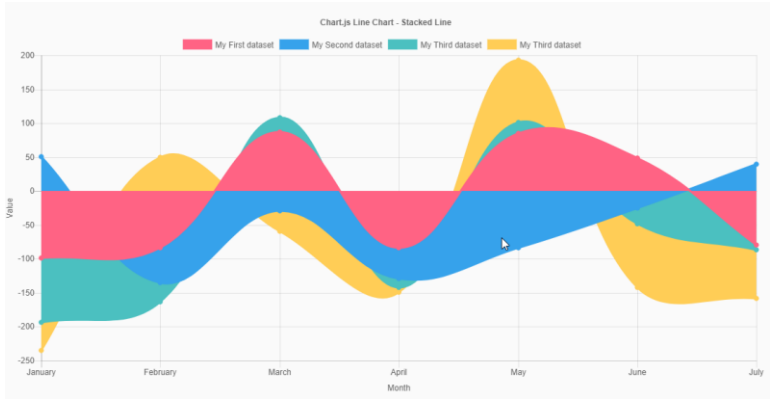
Tamás Kornél

Konzulens: Krébesz Tamás

Feladat ismertetése

- Ismerkedés az egyszerű IoT rendszerekkel - ESP 8266
- Ismerkedés a kapcsolódó (szoftveres) technológiákkal
- Adatgyűjtés
- Adatvizualizálás

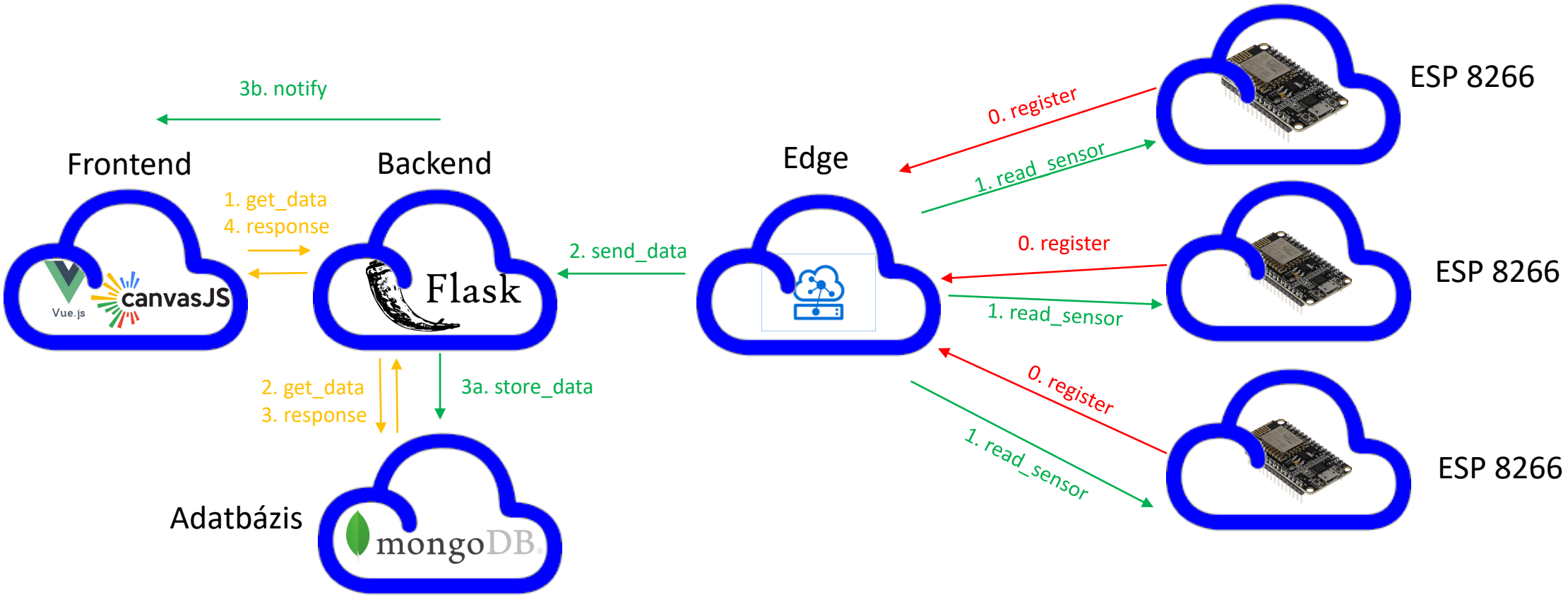




Architektúra

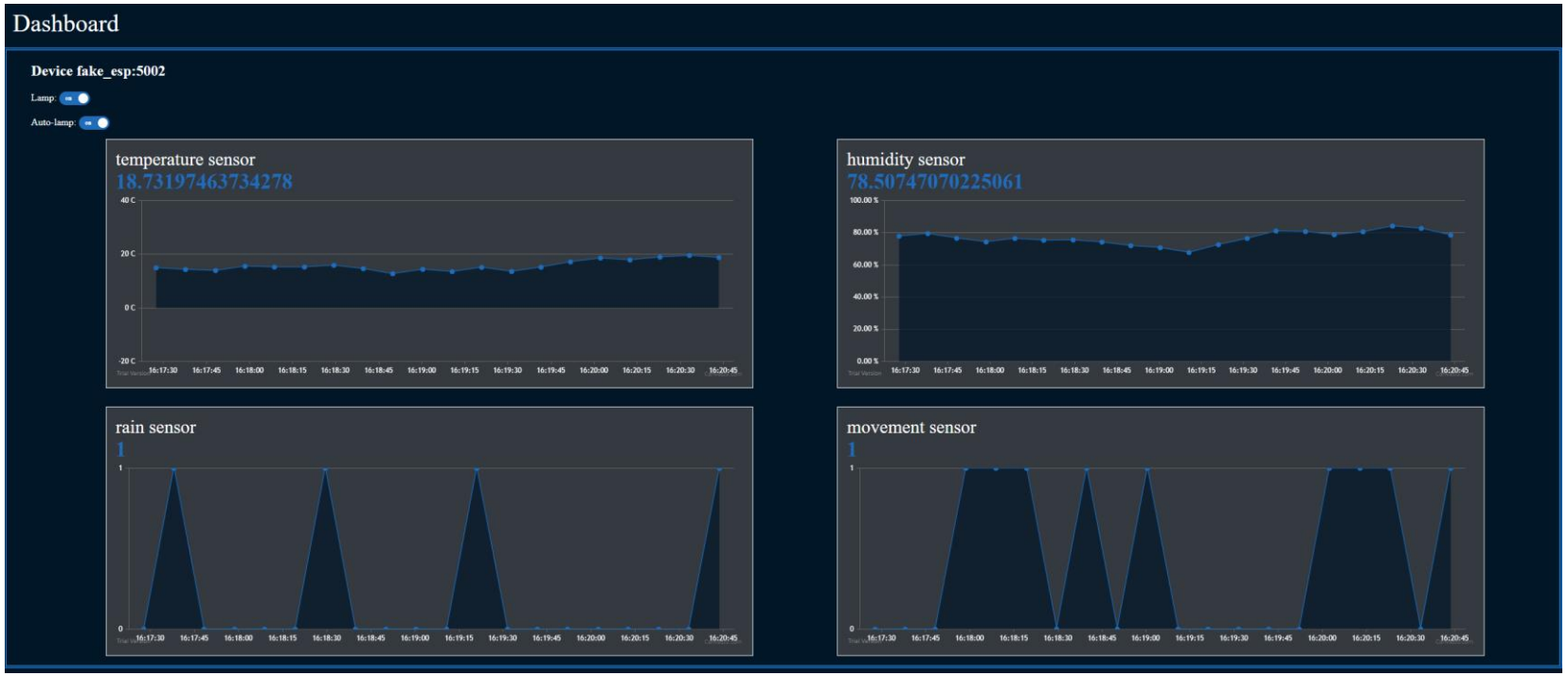
Frontend	Backend	Adatbázis	Edge	Hardver
• Megjelenítés • Interakció	• Logikai műveletek • Transzformációk	• Adattárolás	• Adatgyűjtés • Esp csoportosítás	• Szenzor kiolvasás • Továbbítás

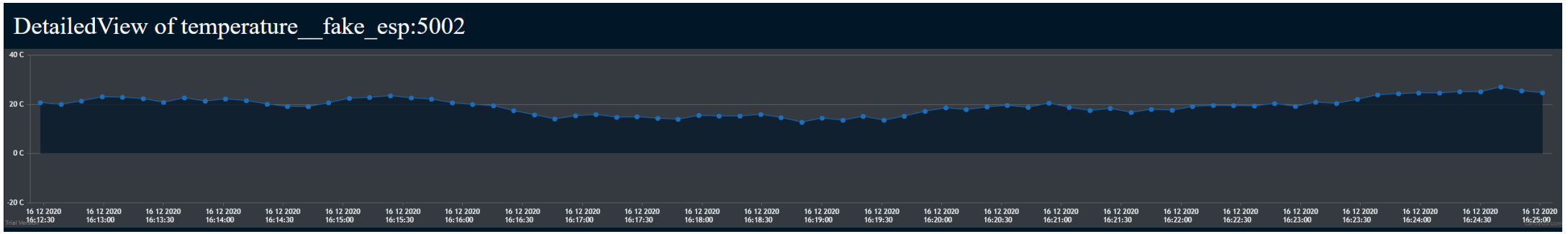
Kommunikáció



Frontend

- VueJS
  - Egyszerű, népszerű → produktivitás++
  - .vue fájlok létrehozhatók
    - <template> → html-szerű váz
    - <script> → javascript ‘turbózva’
      - Két irányú automata adatkötés
      - Templating
      - vue-router
    - <style> →html elemek stílusa
- CanvasJS
  - Külön library
  - Látványos diagramok gyors készítése





Backend

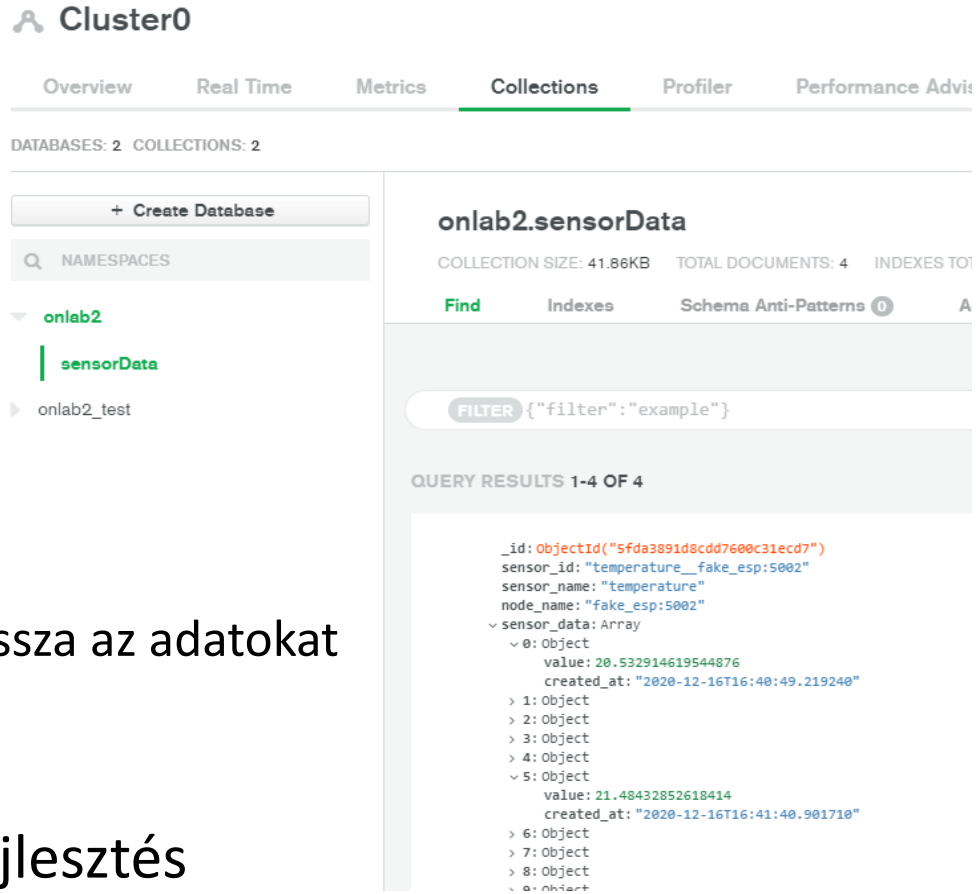
```
app.add_url_rule("/api/sensor", methods=["GET"], view_func=SensorController.get_all_sensor_data)

app.add_url_rule("/api/sensor", methods=["POST"], view_func=SensorController.add_new_sensor)

app.add_url_rule("/api/sensor/set_lamp", methods=["POST"], view_func=SensorController.set_lamp)

app.add_url_rule("/api/sensor/<sensor_id>", methods=["GET"],
                 view_func=SensorController.get_sensor_data_by_sensor_id)

app.add_url_rule("/api/sensor", methods=["DELETE"],
                 view_func=SensorController.delete_every_data_for_every_sensor)

app.add_url_rule("/api/sensor/<sensor_id>", methods=["DELETE"],
                 view_func=SensorController.delete_every_data_for_sensor)
```

- Flask
  - Python
  - Egyszerű, Rapid fejlesztés → produktivitás++
  - Népszerű, könnyen bővíthető
    - pl: *pip install pymongo*
  - Rest végpontok készítésére ideális
  - json adatok továbbítás, transzformációja egyszerű

- Pytest
  - Teszt könyvtár pythonra
  - Könnyen olvasható kód
  - Gyors, hatékony mockolás
    - nem szennyezi a teszt az éles adatbázist
    - A többi komponenstől függetlenül tesztelhető

```
def test_add_new_sensor_with_wrong_format(self, flask_webserver):
    response1 = requests.post(f"{url_base}/api/sensor", json={
        "sensor_id": "sensor3"
    })
    response2 = requests.post(f"{url_base}/api/sensor", json={
        "value": "12"
    })

    assert response1.status_code == 406
    assert response1.text == "required format is: {sensor_id: id, value: value}"

    assert response2.status_code == 406
    assert response2.text == "required format is: {sensor_id: id, value: value}"
```

Adatbázis



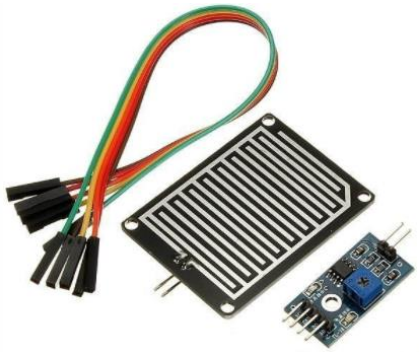
- mongoDB
  - noSQL adatbázis
    - json formában tárol
    - Nem kell transzformálni oda-vissza az adatokat
  - Jól skálázódik
  - Népszerű
  - Nem kér típusokat → rapid fejlesztés
  - Nagyon jó könyvtárak vannak a python integrálásra (pl. pymongo)
- Dolgozhatunk vele
  - Lokálisan, telepítés után
  - Felhőben, ingyenes regisztráció után (cloud.mongodb.com)

```python
def get_sensor_data_by_sensor_id(sensor_id):
    sensor_data = common_config.mongo.db.sensorData
    return json_util.dumps(sensor_data.find({"sensor_id": sensor_id}))
```

Edge

- Ugyanúgy Flask alapú webszerver
- Több esp eszköz ‘kollektora’
- Publisher/subscriber tervezési minta
  - Statikus ip
    - az esp eszközök megtalálják
    - Majd ‘regisztrálják’ magukat
  - Majd az edge minden X. másodpercben lekérdez tőlük
- Beállítható szenzorok (led) változtatása, ha hozzá érkezett kérés

Valós hardver



- Használt szenzorok
  - Hőmérő & páratartalom mérő
  - Eső érzékelő
  - Led szalag
- PlatformIO
  - Népszerű, c++ alapú
  - Jól integrálódik VSCode-dal
  - Hardver közeli függvénykönyvtár
- Megoldás
  - ESP 8266: wifi képes
  - mini-webszerver
    - Várja a beérkező kéréseket
      - Ha érkezik egy pl. GET */temperature*
        - szenzor érték kiolvasása
        - Mért érték visszaküldése a válaszban
    - Egyszerűen szimulálható

Szimulált hardver

- Flask webszerver
- Beérkező kérés esetén
  - Szimulált szenzoradat a válaszban
  - Alap érték / előző érték +- véletlenszerű értékek

```python
def humidity(self):
    newvalue = self.__previous_humidity + random.uniform(-5, 5)
    self.__previous_humidity = newvalue
    if not 0 <= self.__previous_humidity <= 100:
        self.__previous_humidity = 70.0

    return {"value": self.__previous_humidity}
```

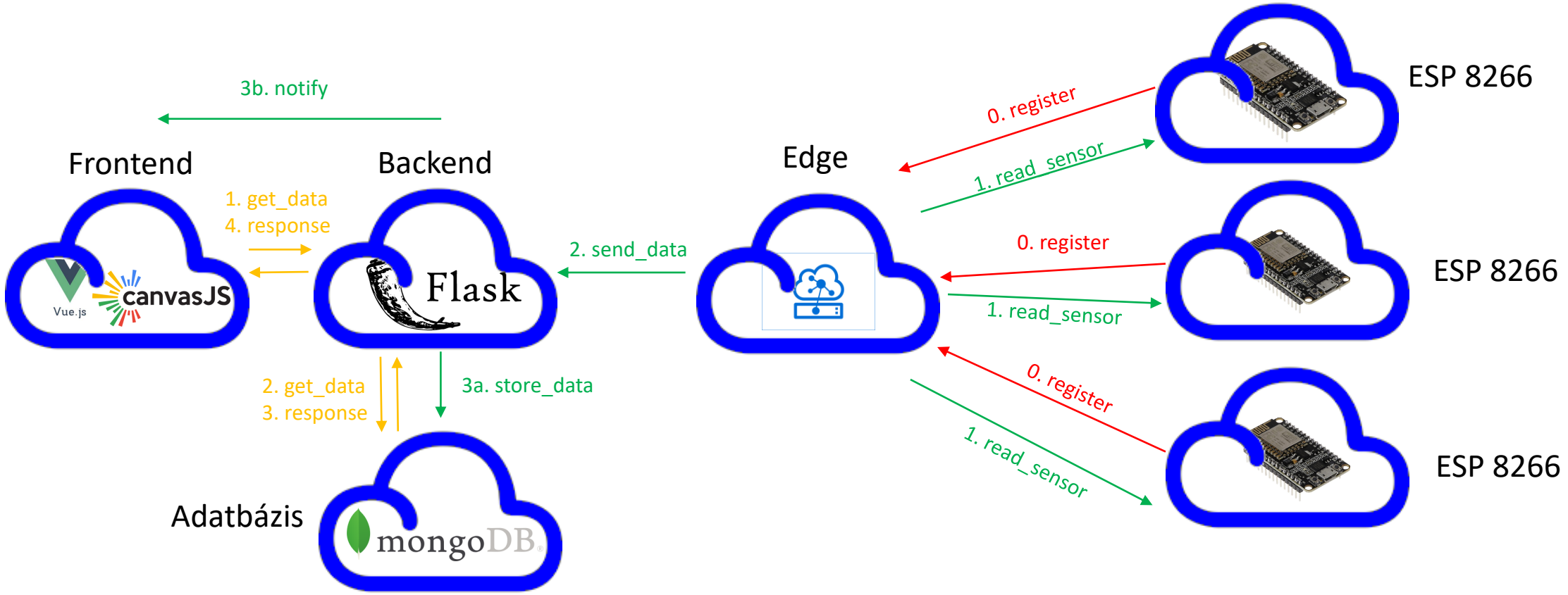
Docker

- Alapötlet: minden komponens egy független kontérenben
  - Linux alapú
  - ipari standard
  - Sokkal gyorsabb, könnyebb, mint egy virtuális gép
  - Közös erőforrások a host géppel
- 5 független konténer
  - Frontend, backend, edge, mongoDB, szimulált esp
- Docker hub
  - Előre elkészített képek – alapként
- Bash script
  - Az összes konténer elkészítése & indítása

```
FROM python:3.8

ARG nodename=backend
ARG nodeport=5000

WORKDIR /$nodename

ENV FLASK_APP=main.py
ENV own_host=$nodename
ENV own_port=$nodeport

COPY ./src /$nodename
RUN pip install -r /$nodename/requirements.txt

EXPOSE $nodeport

CMD ["python", "main.py"]
```

Összefoglalás

- Elosztott rendszer, szenzoradatok:
  - Kiolvasására
  - Megjelenítésére
  - Tárolására



Köszönöm a figyelmet!

Kérdések?