SMART ÜVEGHÁZ BACKEND – TECHNOLÓGIAI DOKUMENTÁCIÓ

1. Bevezetés

Ez a dokumentáció az „Időjárásvezérelt öntözés- és szellőztetésirányítási rendszer” backend komponensének technológiai leírását tartalmazza.
A dokumentum célja, hogy áttekinthető formában bemutassa:
- a rendszer architektúráját,
- a felhasznált technológiákat,
- a csomagstruktúrát és fő komponenseket,
- a futtatáshoz szükséges lépéseket,
- a legfontosabb üzleti folyamatok technikai megvalósítását.

A dokumentáció nem a felhasználói szemszögre, hanem a rendszer fejlesztői / üzemeltetői oldalára fókuszál.

2. Rendszeráttekintés

A rendszer egy Java alapú Spring Boot backend alkalmazás, amely egy vagy több üvegház szenzoradatait fogadja, tárolja és az adatok alapján automatizált vagy félig automatizált vezérlést valósít meg (öntözés, szellőztetés, árnyékolás, világítás, párásítás).

A backend fő feladatai:
- üvegházak és metaadatok (helyszín, növénytípus, aktív státusz) kezelése,
- szenzoradatok fogadása REST API-n keresztül (JSON formátumban),
- időjárásadatok lekérdezése külső időjárás API-ról,
- növényprofilok és öntözési/szellőztetési tervek kezelése,
- szabályalapú döntés az aktuális mérési értékek alapján,
- vezérlőparancsok előállítása,
- naplózás (logikai események, vezérlési döntések).

Perzisztencia szempontjából a rendszer MongoDB dokumentum-orientált adatbázist használ.

3. Architektúra

3.1. Réteges felépítés

A backend egy klasszikus réteges architektúrát követ:
- Controller réteg (REST API végpontok),
- Service (üzleti logika) réteg,
- Repository (perzisztencia) réteg,
- Integrációs réteg (külső időjárás API, gateway-kommunikáció).

Ez a felépítés lehetővé teszi, hogy az egyes rétegek önállóan változzanak, miközben az architektúra logikailag ugyanaz marad.

3.2. Fő komponensek

Logikai szinten a rendszer három nagy funkcionális blokkra osztható:
1) Üvegház-adatkezelés (Greenhouse, növényprofilok, aktív státusz).
2) Szenzor- és időjárásadat-kezelés (SensorData, WeatherSnapshot).
3) Vezérlés és tervezés (ControlEvent, ActionLog).

4. Felhasznált technológiák

- Programozási nyelv: Java (JDK 17 vagy magasabb)
- Keretrendszer: Spring Boot, Spring Web, Spring Data MongoDB
- Adatbázis: MongoDB (dokumentum-orientált NoSQL)
- Build eszköz: Maven
- Konfiguráció: application.yml, YAML-alapú növényprofilok
- Lombok (getter/setter, konstruktor generálás)
- Konténerizáció és futtatás: Docker és Docker Compose

A rendszer fejlesztése és futtatása során Docker Compose használható a backend és a kapcsolódó szolgáltatások (pl. MongoDB) egyszerű indítására, újraépítésére. A legfontosabb parancsok:

Docker indítása (build és futtatás):
docker-compose up --build

Docker tartalmának ürítése és újra buildelése:
docker-compose down -v && docker-compose up --build

Backend API frissítése Dockerben (csak a backend konténer újraépítése):
docker-compose up -d --no-deps --build backend

5. Projektstruktúra és csomagok

A Maven projekt tipikus Spring Boot felépítést követ:

```
src/
 main/
  java/
   hu.nje.smartgreenhouse/
    controller/
    service/
    repository/
    model/
    dto/
    config/
    integration/
    scheduler/
  resources/
   application.yml
   plant-profiles/
    tomato.yml
    cucumber.yml
```

5.1. Controller csomag

A controller csomag REST végpontokat valósít meg, például:
- GreenhouseController – üvegházak kezelése,
- SensorController – szenzoradatok fogadása,
- PlanController – tervek lekérdezése,
- ControlController – kézi vezérlési parancsok fogadása.

5.2. Service csomag

Az üzleti logika a service rétegben található. Az interfészek (pl. GreenhouseService, SensorService) és implementációik (GreenhouseServiceImpl, SensorServiceImpl) gondoskodnak:
- az üvegházak létrehozásáról, módosításáról,
- a legutóbbi szenzoradatok előkészítéséről,
- az időjárásadat lekéréséről és mentéséről,
- a szabályalapú döntések meghozataláról (RuleEvaluatorService),
- a vezérlési események naplózásáról (ActionLogService).

5.3. Repository csomag

A repository csomag Spring Data repository interfészeket tartalmaz (GreenhouseRepository, SensorDataRepository, WeatherSnapshotRepository, PlanRepository, ActionLogRepository, ControlEventRepository, PlantProfileRepository).

A Spring Data képes a metódusnevek alapján automatikusan lekérdezéseket generálni (pl. findByCode, findByGreenhouseCode).

5.4. Model (domain) csomag

A domain csomag a fő dokumentumokat / entitásokat tartalmazza:
- Greenhouse – üvegház,
- SensorData – szenzoradat,
- WeatherSnapshot – időjárási mérés,
- Plan – öntözési/szellőztetési terv,
- PlantProfile – növényprofil,
- ActionLog, ControlEvent stb.

6. Futtatási környezet és telepítés

Előfeltételek:
- JDK 17 vagy magasabb,
- Maven,
- futó MongoDB szerver,
- internetkapcsolat (időjárás API eléréséhez),
- opcionálisan Docker és Docker Compose a konténerizált futtatáshoz.

Build:
- mvn clean package

Futtatás:

Közvetlenül JAR-ból:
- java -jar target/smart-greenhouse-backend-0.0.1-SNAPSHOT.jar

Vagy Docker Compose használatával:
- docker-compose up --build

7. Konfiguráció

Példa application.yml részlet:

```
server:
  port: 8080
```

```
spring:
  data:
    mongodb:
      uri: mongodb://localhost:27017/smart_greenhouse

weather:
  api:
    base-url: https://api.open-meteo.com/
    timeout-ms: 5000
```

A növényprofilok YAML fájlokban tárolhatók a plant-profiles mappában.

8. Fő üzleti folyamatok technikai leírása

8.1. Szenzoradat fogadása

1) A gateway egy PUT /api/sensors/{sensorCode} végpontot hív meg.
2) A SensorController felolvassa a JSON törzset és DTO-vá alakítja.
3) A SensorService menti az adatot a SensorDataRepository segítségével, majd szabályalapú értékelést végez.
4) Szükség esetén vezérlési esemény (ControlEvent) jön létre, és naplózás (ActionLog).

8.2. Időzített állapotlekérdezés

A GreenhouseScheduler @Scheduled metódusa időzített módon meghívja a GreenhouseService.pollAllGreenhouses() metódust, amely minden aktív üvegházhoz lekéri az aktuális időjárásadatokat, és WeatherSnapshot dokumentumként eltárolja azokat.

8.3. Szabályalapú döntési logika

A RuleEvaluatorService a szenzor- és időjárásadatokat a növényprofilban megadott tartományokkal veti össze. Ha az érték kilóg a tartományból, a szolgáltatás akciót javasol (például öntözés indítása vagy leállítása).

9. Naplózás és hibakezelés

A rendszer naplózza a fontosabb bejövő kéréseket, a vezérlési döntéseket, és az időjárás API-hívásokat. Hiba esetén a felhasználó egyszerű, érthető hibaüzenetet kap, a részletesebb információ a logokban érhető el a fejlesztők és üzemeltetők számára.

10. További fejlesztési lehetőségek

Lehetséges fejlesztési irányok:
- autentikáció és jogosultságkezelés bevezetése,
- MQTT vagy más protokoll használata a gateway és a backend között,
- összetettebb szabálymotor,
- történeti adatokra épülő riportok és grafikonok,
- grafikus tervszerkesztő az öntözési és szellőztetési tervekhez.