Backend Developer Challenge

General

We want to supply factories with acceleration sensors. Those sensors can be attached to industrial machines monitoring their vibrations. They will send the data every 100ms to a gateway located in the factory itself using BLE. The gateway streams the data to a server application. A factory should have up to 5 gateways each receiving data from 50 different sensors. In the near feature we estimate to supply up to 1000 factories with this setup.
On the server, the data is analyzed with sophisticated machine learning models providing predictions of the machine state and production quality.
An online application is providing the functionality to manage the sensors, gateways and predictions for a company - per company this tool is used by roughly 20 users.

Java/Backend Challenge

Technologies to use:

- Java 8 or Java 11
- Maven or Gradle
- Spring Boot - latest
- PostgreSQL - latest
- Hibernate
- optional: Flyway database migration
- optional: Docker
- optional: Docker Compose
- feel free to add any required dependency

This Java application is the management-service described above, used to manage machines, sensors, gateways and predictions.
In this challenge, we want you to implement a simple java backend using the technologies listed above.
The result should be an executable .jar or docker-compose file linking to the correct images. In addition the source code should be located in a (public or private) git repository and accessible for us.
The application is a basic CRUD application where the user can add and edit basic information about machines in factories.
The Java backend application should implement the following features:

- Machine should be persisted in database and should have at least the following fields:
  - id (uuid or serial)
  - createdAt
  - updatedAt
  - name

- REST endpoints for communication by the frontend
- For communication separate DTOs should be used (not JPA entities directly)

REST-Endpoints

The following CRUD-REST Endpoints compliant with the REST-Maturity-Model Definition by Martin Fowler (https://martinfowler.com/articles/richardsonMaturityModel.html) should be available at REST-Level 2:

- fetch all machines (ordered by last modified first)
- fetch a single machine
- create a single machine
- update a single machine
- delete a single machine (soft-delete - keep the physical record in database!)