Backend Engineer - Coding Tasks

Abstract

The task is intended to expose the candidate to small-scope real-life design and development challenges.

Assumptions

1. You can use any backend Java framework you like e.g. Spring Boot, J2EE, J2EE Microprofile, OSGI (god forbid ;))
2. If you need a DB, please use (for the task) something embedded like H2, HSQLDB, or Derby.
3. You can (and should, if possible) use third-party libraries to leverage parts of the functionality that are “generic.”

Domain Background

Vocabulary

A **Treatment Plan** describes the type and schedule of the treatment actions for a particular patient and has the following properties:

1. Treatment action: enum of possible actions [ActionA or ActionB]
2. Subject patient: reference to the subject patient
3. Start time: timestamp of earliest possible treatment task
4. End time: timestamp of the latest possible treatment task or empty if the plan is endless
5. Recurrence pattern: for example, “every day at 08:00 and 16:00” or another example, “every Monday at 12:00”

A **Treatment Task** describes a single treatment action to be done for a specific patient at a specific time and has the following properties:

1. Treatment action: enum of possible actions [ActionA or ActionB]
2. Subject patient: reference to the subject patient
3. Start time: timestamp of the earliest time the task can be completed
4. Status: Active or Completed

Functional Requirements

1. [Given, outside of scope, no need to implement] As a Clinician, I can create a treatment plan for a specific patient so I can communicate to all the treatment tasks executor what has to be done
2. [Given, outside of scope, no need to implement] As a Nurse, I can browse/view treatment tasks so I can execute treatment actions on time and report their completion
3. **[Need to be implemented]** As a System, I can generate treatment tasks based on the treatment plan
    a. Your goal is to implement a “scheduling” execution service that “spawns” treatment tasks according to all the treatment plans existing in the system and their schedule (start time, recurrence pattern, and end time)
    b. We can **start with simple implementation** and then discuss the non-functional requirements like scalability and resilience.
    c. Software Design Hints
        i. Treatment Plans and Treatment Tasks are stored in separate tables in DB, but you decide on the schema/layout as needed.
            1. Reference to the patient is a simple string
            2. The entities are denormalized, and this is OK
            3. You can add any field you think is needed.
        ii. Treatment Plan’s insertion into DB is outside of the scope of this task, and you can assume that they are already there
        iii. The result of this coding task is a scheduler that creates new Treatment Tasks stored in DB according to what is defined in active Treatment Plans it can find in DB. No need to execute the tasks; just store them in DB.
        iv. You should not use the Quartz scheduler as a whole; you are supposed to design and implement the “executor” part of its functionality.

FAQ and Comments:

1. Should I support human-readable recurrence patterns? - Recurrence patterns format is an implementation detail and is not required to be humanly readable
2. Should I create all or some of the treatment tasks ahead of time? - No need. You need the treatment task to exist in the DB when the task is active. Creating too many tasks ahead of time can be a scalability issue.