PANDEMIC COMBAT

05 de Setembro de 2021
Documentação do Sistema Pandemic Combat

Adeildo Lucas Guerra Pereira
adeildo.pereira@ccc.ufcg.edu.br

Sumário

Apresentação do Problema: 3
Requisitos (MVP 1) 3
Notas: 4
Apresentação da solução: 5
Introdução: 5
Implementação: 5
Diagrama de classes do sistema: 7

Apresentação do Problema:

Quando o mundo é atingido por uma pandemia sem remédio imediato, além das habilidades dos profissionais de saúde, é preciso ter um bom sistema de informações para ajudar nas tomadas de decisões, amenizar ao máximo seus impactos.

Assim, ainda que não seja da área de saúde, você pode ajudar no combate. Para isso, foi designado para desenvolver um sistema que irá coletar informações de todo país, organizá-las e prover informações com base nelas.

Requisitos (MVP 1)

Você irá desenvolver uma **API RESTFul** (a ideia é que facilmente outros sistemas consigam se integrar para prover e obter dados), ao qual irá coletar informação sobre os hospitais, seus recursos (pessoais e materiais), pacientes em atendimento (versões futuras), etc., ajudando no intercâmbio de recursos.

- **Adicionar hospitais**

  Um hospital deve ter um *nome*, *endereço*, *CNPJ*, *localização* (latitude, longitude, etc.).

  Ao adicionar o hospital, junto deve ser adicionado seus recursos atuais bem como seu percentual de ocupação.

- **Atualizar percentual de ocupação de um hospital**

  Um hospital deve poder reportar seu percentual de ocupação a todo instante, de forma que isso possa ser usado no processo de intercâmbio de recursos.

- **Hospitais não podem adicionar / Remover recursos**

  Os recursos dos hospitais só podem ser alterados via intercâmbio. Aquisição de recursos avulso será feita em outra API, pois requer um processo específico.

- **Intercâmbio de recursos**

  Os hospitais de trocar recursos entre eles.

  Para isso, eles devem respeitar a tabela de valores abaixo, onde o valor do recurso é descrito em termos de pontos.

  Ambos os hospitais oferecerem mesma quantidade de pontos. Por exemplo, 2 respiradores e 1 enfermeiro (2 x 5 + 1 x 3 = 13), valem o mesmo que 1 médico e 1 ambulância (1 x 3 + 1 x 10 = 13). Esta regra pode ser quebrada caso algum hospital esteja com ocupação maior que 90%, onde ele poderá oferecer menos recursos que outro hospital no intercâmbio.

  Além de colocar uma negociação por questões de histórico, os itens devem ser transferidos de um hospital a outro.

Item	Pontos
1 médico	3 pontos
1 Enfermeiro	3 pontos
1 respirador	5 pontos
1 tomógrafo	12 pontos
1 Ambulância	10 pontos

- **Relatórios**

  A API deve oferecer os seguintes relatórios:

  1. Porcentagem de hospitais com ocupação maior que 90%.
  2. Porcentagem de hospitais com ocupação menor que 90%.
  3. Quantidade média de cada tipo de recurso por hospital (Ex: 2 tomógrafos por hospital).
  4. Hospital em superlotação (ocupação maior que 90%) a mais tempo.
  5. Hospital em abaixo de superlotação (ocupação maior que 90%) a mais tempo.
  6. Histórico de negociação.

Notas:

1. Deverá ser usado Java, Spring boot, Spring Data, Hibernate (pode ser usado o banco de dados H2) e como gerenciador de dependência Maven ou Gradle.
2. Não será necessária autenticação.
3. Nós ainda nos preocupamos com uma programação adequada (código limpo) e técnicas de arquitetura, você deve demonstrar isso mesmo em meio a uma pandemia.
4. Não esqueça de documentar a sua API.
5. Sua API deve estar minimamente coberta por testes (Unitários e / ou integração).
6. Da descrição acima você pode escrever uma solução básica ou adicionar requisitos não necessários. Use seu tempo com sabedoria; uma solução ótima e definitiva pode levar muito tempo para ser efetiva, então você deve trazer a melhor solução possível, que leve o mínimo de tempo, mas que ainda seja capaz de demonstrar suas habilidades.
7. Comente qualquer dúvida e cada decisão tomada.

Apresentação da solução:

Introdução:

Com o intuito de resolver o problema proposto, foi pensado em um sistema com uso de Java, Spring boot, Spring Data, Hibernate – banco de dados H2 e como gerenciador de dependências foi usado o Maven. Para acesso as funcionalidades do sistema foi utilizado a ferramenta Swagger.

Implementação:

O sistema está sendo totalmente acessado via Swagger no link:

http://localhost:8080/swagger-ui.html#/

Para o controle e acompanhamento das tabelas e relações do banco de dados H2, usa-se o link:

http://localhost:8080/h2

Para implementar as funcionalidades solicitadas foi pensado em três classes principais, Hospitais, Recursos e intercambioDeDependencias. Cada hospital possui seus recursos e também um nome, endereço, CNPJ e localização como foi especificado, além disso um atributo do tipo Date foi adicionado junto com um do tipo boolean (status de ocupação). Esses atributos servem para o controle de tempo em que um hospital fica em alta ou baixa ocupação.

O sistema foi dividido todo em pacotes, controller, model, DTO, service e repositor seguindo o padrão spring. Existem dois controllers, um para Hospitais e um para Recursos, as classes DTO auxilia na adição de novos hospitais, a mesma guarda todas as informações de atributos necessários para criar uma nova entidade, sendo passada como parâmetro para requisição (@RequestBady) e apresentando todo escopo de informações necessárias para determinada função.

No que se refere a service, nela contém todas as interfaces de comunicação com os controllers e a implementação da lógica do sistema. Já as classes presentes no pacote repository as mesmas são utilizadas exclusivamente para interação do sistema com o banco de dados. Realizando consultas e adicionando novas tabelas a partir delas.

Quando se adiciona um novo hospital ao sistema, é feita uma verificação de porcentagem de ocupação, a partir disso o atributo boolean citado acima será setado para true caso a ocupação seja maior ou igual a 90% e para false caso contrário. Já o Date, o mesmo é criado no momento da criação da nova entidade no sistema, ele servirá para verificação de tempo. Uma das funcionalidades do sistema é a atualização da ocupação, neste caso, quando a ocupação é alterada e seu status muda a data é modificada para que seja possível verificar tempo em que o hospital se encontra nesse novo status. Desta forma, foi possível garantir a verificação e controle de relatórios exigidos.

Para funcionalidade de intercambiar recursos, foi pensado em uma requisição que em seu corpo, traz informações referente aos hospitais envolvidos, e os recursos que

serão intercambiados. No momento da solicitação uma classe útil é acionada para garantir que a regra de intercambio seja cumprida:

Ambos os hospitais devem oferecer a mesma quantidade de pontos.

Item	Pontos
1 médico	3 pontos
1 Enfermeiro	3 pontos
1 respirador	5 pontos
1 tomógrafo	12 pontos
1 Ambulância	10 pontos

Além disso, novas regras se aplicam quando um dos hospitais envolvidos na requisição está com alta ocupação.

Todos os relatórios estão sendo gerados a partir de consultas ao banco de dados, e organização das informações obtidas nas mesmas.

Diagrama de classes do sistema: