Welcome to Koerber Pharma Backend Challenge

You work in your city's hospital. Among other responsibilities, you need to create a way to log the consults that happen in the hospital so you can keep track of patient's consults, their pathologies and respective symptoms. The hospital board wants to also be able to keep adding new consults and patients every day and for now they are happy with the pathologies and symptoms they have in their database so that is out of scope.

The hospital currently offers 5 different specialties:

- Dermatology
- Ophthalmology
- Radiology
- Family Medicine
- Pediatrics

For this problem, the following assumptions are true:

- Each consult has a doctor, a specialty and a patient.
- For the sake of simplicity let's assume that there's only one doctor per specialty.
- A Patient can go to multiple consults in different specialties and can even go to the same specialty more than once.
- Each patient can have multiple pathologies that it suffers from.
- Each pathology can have multiple symptoms.

In the following table you can find all the data that you will need to have in your database for this exercise. Although our goal is mostly to analyze your thought process and coding skills please **DO NOT** change this data so we can provide quicker feedback.

Consult	Doctor	Specialty	Patient (Age)	Pathology	Symptom
1	António	Dermatology	Manuel (53)	Pathology 1	Symptom 1 Description
					Symptom 2 Description
2	António	Dermatology	Manuel (53)	Pathology 2	Symptom 3 Description
					Symptom 4 Description
					Symptom 5 Description
3	Maria	Ophthalmology	Manuel (53)	Pathology 3	Symptom 6 Description
					Symptom 7 Description
4	Maria	Ophthalmology	Joana (32)	Pathology 3	Symptom 6 Description
					Symptom 7 Description
5	Carlos	Radiology	Ana (25)	Pathology 4	Symptom 8 Description
6	Gabriela	Family Medicine	Diogo (33)	Pathology 5	Symptom 9 Description
					Symptom 10 Description
					Symptom 11 Description
7	Paulo	Pediatrics	Catarina (33)	Pathology 6	Symptom 12 Description
					Symptom 13 Description
8	Maria	Ophthalmology	Miguel (40)	Pathology 7	Symptom 14 Description
					Symptom 15 Description

Objectives

Your backend will be exposed through HTTP server and will contain 4 endpoints that a frontend could use in the future to control this hospital's consults and patients. These endpoints should allow the following:

Create consults

This endpoint should allow a front-end to connect to the API and be able to register new consults that happen in the hospital.

Get patient consults and symptoms

For a specific patient, please design an endpoint that can return a list of all the consults that the patient has had and also all the symptoms he has presented in each consult.

An example response could be:


```
{
  "Consults": [
    {
      "ConsultId": 1,
      "Doctor": "António",
      "Specialty": "Dermatology"
    },
    {
      "ConsultId": 2,
      "Doctor": "António",
      "Specialty": "Dermatology"
    },
    {
      "ConsultId": 3,
      "Doctor": "Maria",
      "Specialty": "Ophthalmology"
    }
  ],
  "Symptoms": [
    {
      "SymptomId": 1,
      "Description": "Symptom 1 Description"
    },
    {
      "SymptomId": 2,
      "Description": "Symptom 2 Description"
    },
    {
      "SymptomId": 3,
      "Description": "Symptom 3 Description"
    },
    {
      "SymptomId": 4,
      "Description": "Symptom 4 Description"
    },
```

```
    {
      "SymptomId": 5,
      "Description": "Symptom 5 Description"
    },
    {
      "SymptomId": 6,
      "Description": "Symptom 6 Description"
    }
  ]
}
```

Get top specialties

Looking at all the specialties that the hospital has to offer, the directors want to know which specialties are having more than a certain number of patientes. With this in mind, this endpoint should allow us to know which specialties have more than 2 UNIQUE patients.

An example response could be:

```
{
  "SpecialtyName": "Ophthalmology",
  "NumberOfPatients": 3
}
```

Get all patients

This endpoint will return the data for all the patients with pagination, filtering and sorting by either age or name. This means that this endpoint will accept several parameters (Either GET or POST).

Technical Requirements

- Java Spring boot
- Unit Tests
- Performance tests (Bonus points)
- Open Api 3
- ORM
  - Migration Strategies
- Clear curl instructions
- Docker
  - The provided docker should be updated or you can build one new as long as the application runs with a simple command. This includes tests and database population.
- Centralized logging using containers (**Hint**: don't come up with your own logging solution. Use something that runs out of the box and you can quickly add to the project.)

What we're looking for

- Clean project setup and well documented

- Elegant/fast way to upload data
- Ability to dive into a new topic, extract the important points and then code it up.
- Make it snappy
- Document your approach, your decisions and your general notes
- Fulfill the exercise expectations.
- OpenAPI on REST endpoints.
- Write clear and well-structured code. You won't work alone. It should be easy to understand and modify your code.
- Spend no more than 2/3 hours. We estimate that this exercise could be done in 2/3 hours. You already have things to do in your own time and we don't want this exercise to alter too much your life.
- Send working instructions. Imagine that you bought a product that you need to install by yourself and when you follow the instructions provided, it does not work. It would be frustrating.
- Use technologies you know. This is not time to test something new unless you are really sure about it. Use technologies you already know and have experience with.
- **We expect a README file in the repository.** This README must explain very clearly the instructions for us to start the application without any type of debugging or complication. The instructions must work out of the box. **This is one of the most important part of the problem.** You need to think always about who is going to use it and try to reduce frustration.
- **Share the repo** (Github/Gitlab/Bitbucket) Any code you write must be committed to a Git repository. Please, **start committing since the beginning.** Don't wait to have the entire exercise and just send one commit with all the code. We would like to see your progress.

We don't expect you to

- Be a craftsman. We don't expect to see the most optimized code. Be pragmatic with the time you have and the things you need to do.
- Implement a production service. Forget about authentication, users, SSL, ... Just focus on the functionality described above.
- Implement the most performant solution. There will be time to go over your decisions and discuss the approaches taken.

Have fun!