Task: Implement a sample FileService REST-API & Implementation

Motivation

As part of the distributed platform that we are currently establishing, we want to be able to provide various (micro-)services on different nodes that interact with each other.
One of those serves a large number of files from a distributed file storage backend.

Other services use this service as their file backend so that they are able to store various data centrally. Also, the file service is meant to be a distributed service, so other file service nodes could take over responsibilities in case the first node goes down.

While it's certainly possible to access a distributed file storage directly without a service in-between, it would require to use either client libraries, rely on the API stability of a thirdparty and/or would require extending functionality for unsupported features such as mandatory access control.
Therefore serving files from the storage using a REST service seems to be a viable vendor agnostic alternative.

Task specification

In this task we want you to do a demo implementation of a small subset of that functionality, namely a part of the REST-API that other services can use to perform file operations.

The task is not necessarily meant to be completed by you, it's more important that we can see how you would work on this task and show your skills, as it includes designing the API itself, the implementation of said API and possibly stating decisions if needed.

It's possible to document your API design and decisions in code, but you can also provide other documents if you feel this might be better here.

We prepared both a DropWizard and a Spring Boot Maven project that can serve as the base to the implementation part of the task. Choose one of them as you wish - we are using the first instead of Spring internally in our projects, but there is no requirement to use it. You can modify the selected project in any way that you feel is needed, e.g. changing names, adding/removing modules etc. If you want to add additional libraries, you are also free to do so. You are in charge of the technologies unless specified otherwise.

The project is meant as a quickstart to give you something to start with, but you can also opt for using a completely different bootstrap project for the implementation. If you want, you can use plain Jersey or some other JAX-RS implementation instead.

Develop the REST-API to perform various file operations. Each operation might use the provided backend service (dummy) implementation to perform the task. If you want to add (or change, or remove) functionality to the backend service, you are free to do so. You can also opt to use a different backend for the functionality.

It's not needed to expose every function of the backend service as REST API, you also should decide which one is relevant and how to expose the functionality, e.g. what data types or formats

suit the functionality of a file service best, tough basic CRUD operations should be in a working state. It's also worth thinking about how you would try to make the service scaleable, e.g. how you would make it handle a somewhat arbitrary number of uploaded files or users if that would be a requirement. You also should think about the fact that end users might want to use the API directly, so you should be consistent regarding API usage, validation and return values.

Also, provide some tests that show how you would do unit-tests and client/server integration tests. It's not required to provide a certain test coverage, it's sufficient to show how you would address this requirement.

Additionally, you should provide a REST client that shows the API usage to allow to see how you envisioned the API to be used in other applications.

You should also think about authentication and authorization when designing the API, but for the sake of simplicity of the task, it would be enough to document how you would address this in a distributed environment or provide a dummy implementation.

Each demo project consists of three subprojects - the server application, the client and a shared (common) project for both client and server. The server application is compiling as a standalone application, so that it could be deployed using Docker later (which we are doing in our products). Both the common and client project are intentionally left blank so that you can decide on how they should look like. In the end, there should be various operations that potential consumers of the service can use to interact with the file storage using the client code.

For the server, the FileServiceResource/FileServerController class is meant to be a starting point for the actual server API implementation. The server main class FileServiceApplication will boot up a server on port 8080 and serve up the API there.
The backend service does allow to specify a session id which is meant to be passed to the backend and make the backend action user dependent (which is not implemented in the dummy for the sake of simplicity).

If you opt for using another bootstrap project, you should provide roughly the same project structure, maybe reusing the backend code, but otherwise you can use any technology that allows you to show your skills designing and programming this API.

Good luck!