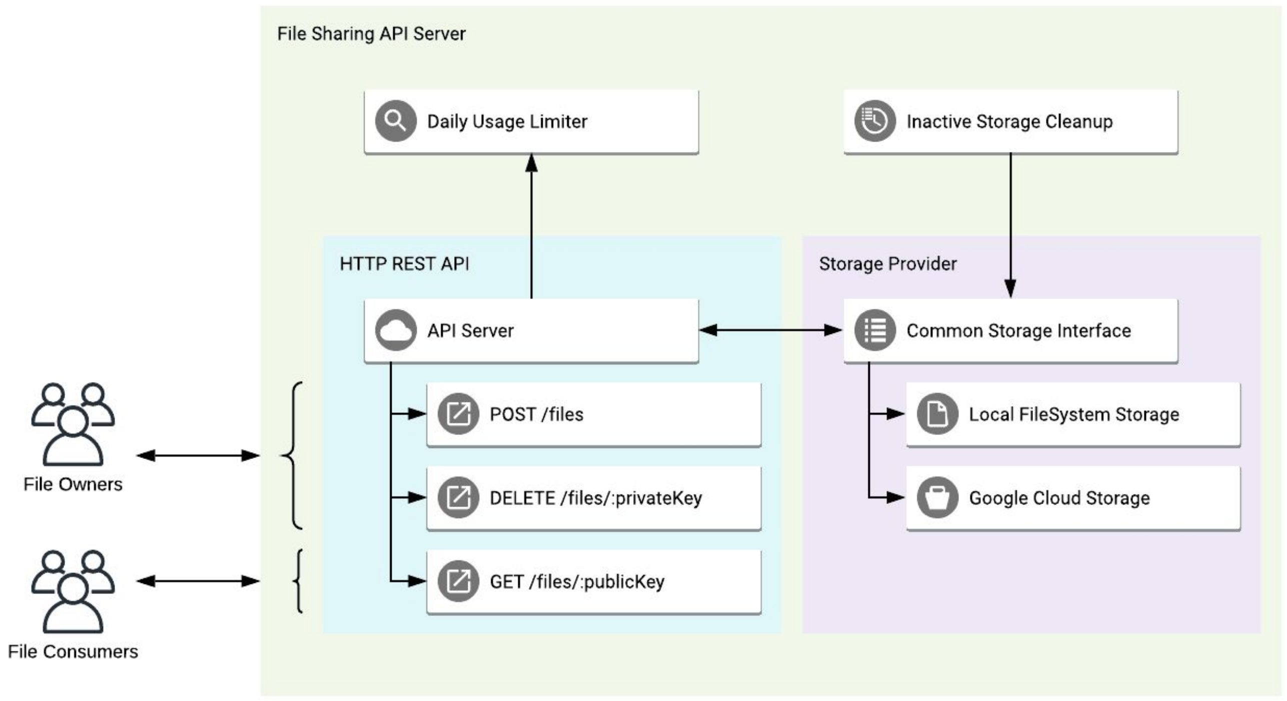
Node.js Backend code test

Requirements

- All code should be commented
- All code should be written in JavaScript (except configuration files)
- The project should run with the latest Node.js LTS - https://nodejs.org/en/download/

If one or more of the features cannot be implemented - please provide your research results (what have you tried and why do you think it cannot be implemented?)

Diagram



Task 1: File Sharing API Server

Note: please use the Diagram above for a reference.

Prerequisites:

1. It should be possible to start the API server using the "**npm start**" command https://docs.npmjs.com/cli/start
2. It should be possible to run unit and integration tests using the "**npm tes**t" command https://docs.npmjs.com/cli/test
3. The following environment variables should be supported:
    a. "**PORT**" - the port number to listen to
    b. "**FOLDER**" - the absolute path to the root folder with all the files (see below)

Requirements:

1. The API Server should implement the following HTTP REST API endpoints:
    a. "**POST /files**" - this endpoint will be used to upload new files. It should accept "**multipart/form-data**" requests and return a response in JSON format with the following attributes: "**publicKey**", "**privateKey**".
    b. "**GET /files/:publicKey**" - this endpoint will be used to download existing files. It should accept "**publicKey**" as a request parameter and return a response stream with a MIME type representing the actual file format.
    c. "**DELETE /files/:privateKey**" - this endpoint will be used to remove existing files. It should accept "**privateKey**" as a request parameter and return a response in JSON format confirming the file removal.
2. All the file access functionality should be implemented as a separate component
    a. This component should encapsulate all the internal file processing details and provide a simple interface for all the actions.
    b. The default implementation should work with local files located inside a root folder defined in the "**FOLDER**" environment variable.
    c. It should be possible to implement other storage providers connected to the popular cloud APIs using the same interface. Examples of such providers: Google Cloud Storage, Microsoft Azure Storage or AWS Cloud Storage.
3. The API Server should implement configurable daily download and upload limits for the network traffic from the same IP address
4. The API Server should have an internal job to cleanup uploaded files after configurable period of inactivity
5. All the HTTP REST API endpoints should be covered by integration tests
6. All the individual component methods should be covered by unit tests

Task 2 (optional): Google Cloud Storage provider

Note: please use the Diagram above for a reference.

Prerequisites:

1. The Google Cloud Storage provider should be implemented using the official library https://www.npmjs.com/package/@google-cloud/storage
2. The following environment variables should be supported in this mode:
   a. "**PROVIDER**" **-** one of the provider types: "**google**", "**local**". Default value **"local"**
   b. "**CONFIG**" **-** the absolute path to the provider configuration file (includes storage credentials, bucket information, etc.)

Requirements:

1. The interface of the Google Cloud Storage provider component should be identical to the interface of the previously implemented local filesystem provider (see Task 1).
2. There should not be any hardcoded provider configuration options, everything should be configurable via the single configuration file provided in the "**CONFIG**" environment variable
3. The Google Cloud Storage provider configuration file should be documented and should be trivial to use