# JFrog

## Improvements

1. I will run over all the URL's and sum all the total\_count, will check in the header if I have

Links in the header, if I have links I will go over all links and append all the items from all the pages.

1. The project will have the following folder structure:

* DTO (Data Transfer Objects) - DTOs are used for the transfer of data between services, encapsulating the request and response data structures. They make it easier to define the structure of the data that is sent over the network.
* Entity - Entities define the structure of data within the database.
* Exception Handling - A global exception handler would be used to manage exceptions across the application, making sure that common errors (like validation errors or service unavailability) are handled uniformly. We will use @ControllerAdvice to globally catch exceptions in Spring applications, making sure to send consistent error responses.
* Repository - Repositories handle database interaction via Spring Data JPA. They abstract away the complexities of database queries and allow you to perform basic CRUD operations (Create, Read, Update, Delete). You will use the JpaRepository interface for basic operations and add custom methods when needed.
* REST Controllers - Controllers define the REST API endpoints that accept HTTP requests and return HTTP responses. They map incoming HTTP requests to the service layer methods.
* Service Layer - The service layer contains business logic and coordinates between the controllers and repositories. This layer is responsible for the actual processing of data and implementing the core functionality.

1. We will do validation for the data, and will throw exception accordingly.
2. SLF4J (Simple Logging Facade for Java) will be used for logging across the application.
3. Swagger 3 is used to generate API documentation and provide interactive endpoints. This allows the developers and users to explore the API via a user-friendly UI.
4. JUnit5 will be used for writing unit and integration tests across the application layers. Tests will ensure that each component works as expected and help prevent regressions.

The tests will be part of the continuous integration (CI) pipeline to ensure that changes do not break existing functionality.