**Spring Boot - 2**

1. Checkout to the branch “spring-boot-app-v1” you created in spring boot - 1
2. Create a new branch “spring-boot-app-v2”
3. Propose a proper directory structure for your project created in spring boot part 1
   * Use nouns instead of verbs: API endpoints should be nouns instead of verbs. This makes it clear what the endpoint does, for example: **/users instead of /get-users.**
   * Use plural nouns: Use plural nouns for collections of resources, for example: **/users instead of /user.**
   * Keep endpoints short and simple: Make sure that endpoints are short and simple, avoiding complex or nested paths.
   * Use lowercase: Use lowercase letters for endpoint names and use hyphens to separate words.
   * Be consistent: Be consistent with the naming convention used throughout the API.
   * Versioning: Use versioning in endpoint names to indicate changes to the API. For example, /v1/users.
   * Use relevant query parameters: Use query parameters to filter or sort resources when necessary.
   * Keep endpoints intuitive: Endpoints should be intuitive and easy to understand for developers using the API.
   * Consider using descriptive names: Consider using descriptive names for endpoints that convey the purpose of the endpoint, for example: /users/search

“RESTful URIs should refer to a resource that is a thing (noun) instead of referring to an action (verb) because nouns have properties which verbs do not have – similar to resources have attributes.” – [RESTfulAPI.net](https://restfulapi.net/resource-naming/)

1. Use API naming conventions to your endpoints
2. Use separate request and response types
3. Use exception handling where applicable

**ResponseEntity is a class in the Spring Framework that represents an HTTP response. It allows you to specify the HTTP status code, headers, and body of a response. This makes it easy to return custom responses to clients based on different error or success scenarios.**

**Before**

Text

Description automatically generated

**After**

**Text

Description automatically generated**

**After**

**Text

Description automatically generated**

1. Return proper status codes
2. Use optionals where possible

* The purpose of Optionals in Java is to provide a way to represent optional values, meaning values that may or may not be present. **This can be useful in many scenarios, such as when you want to avoid null references,** handle missing values in a clear and concise way, or simplify error handling.
* In Java, **null references can lead to unexpected NullPointerExceptions**, which are often difficult to debug and can cause serious problems in your code. By using Optionals, you can avoid null references by explicitly representing the absence of a value.
* Additionally, Optionals provide a clean and concise way to handle missing values by providing methods such as orElse and orElseThrow that allow you to specify what should happen if a value is not present. orElseThrow() is a method in Java that is used to return the value of an Optional object, or throw an exception if the value is not present.

1. Why do we need logging in our applications?

**Logging is an important aspect of any software application, it helps in tracking the flow of execution, debugging errors, monitoring performance and improving the overall stability of the application.**

1. What are different types of log levels?
   1. **TRACE**: Detailed information for debugging purposes, typically not used in production.
   2. **DEBUG**: Information used in debugging the application, including debugging messages and detailed information about method invocations.
   3. **INFO**: General information about the application, such as the application starting up, or a significant event has occurred.
   4. **WARNING**: An abnormal condition has occurred that might cause the application to malfunction.
   5. **ERROR**: An error has occurred, the application may still continue to function, but some functionality may be impacted.
   6. **FATAL**: A critical error has occurred, the application will not continue to function and must be addressed immediately.
2. Use a logging framework and add logs to your application

Text

Description automatically generated

Text

Description automatically generated

1. Push your updated project to “spring-boot-app-v2”
2. Add your codes and answer sheet to a directory named “spring-boot-basic-training-v2” and push it to your training github repository
3. Create a pull request to main branch and assign it to your trainer

**Resources**

<https://nordicapis.com/10-best-practices-for-naming-api-endpoints/>

<https://restfulapi.net/resource-naming/>

<https://www.geeksforgeeks.org/spring-boot-code-structure/>