**Employee Management System - Implementing Pagination and Sorting**

**Scenario:** Add pagination and sorting capabilities to your employee search functionality.

* **Pagination:**

Pagination in Spring Boot allows you to split the results into manageable chunks, making it easier to navigate through large sets of data. You can implement pagination using *Page* and *Pageable* interfaces provided by Spring Data JPA. The Pageable object typically contains the page number, the number of records per page, and the sorting criteria.

When you use pagination, your query returns a *Page* object, which contains the content for the current page along with additional metadata like the total number of pages, total elements, and whether it's the first or last page.

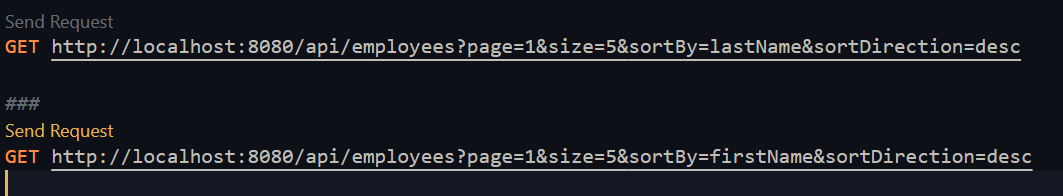
* **Sorting:**

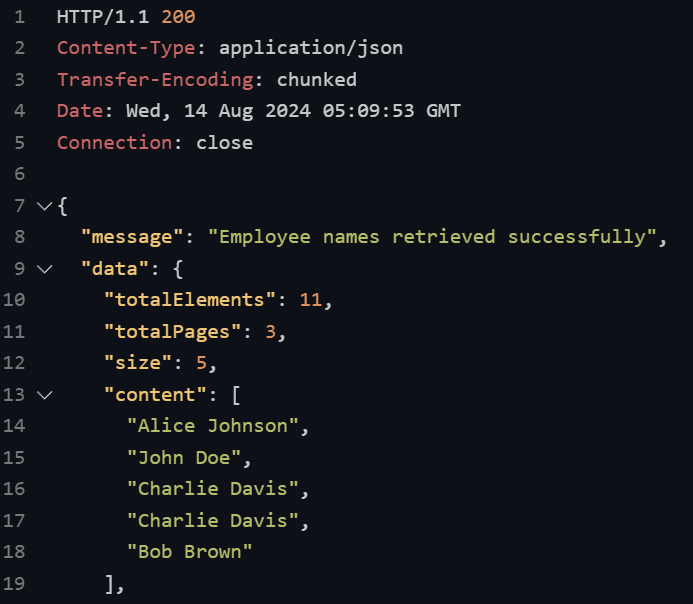
Sorting lets you order your results based on one or more fields (e.g., first name, last name). In Spring Boot, sorting is often combined with pagination. The Pageable interface allows you to specify the sorting criteria along with pagination details. You can sort in ascending or descending order based on the field(s) you choose

* **Combining Pagination and Sorting:**

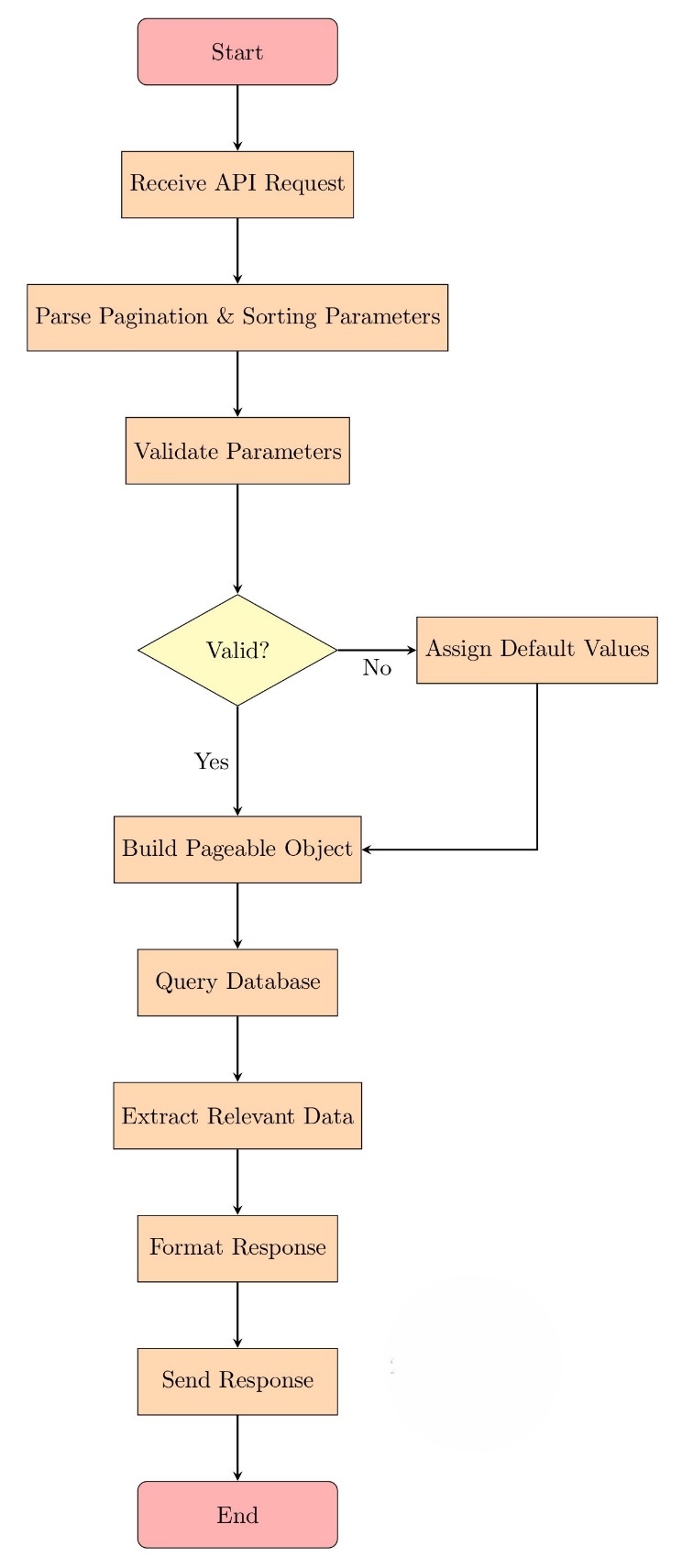
By combining pagination and sorting in your search endpoint, you can provide a user-friendly experience where users can navigate through sorted lists of employees efficiently. The sorting criteria can be adjusted dynamically based on user input.

* **The Output:**

To see the output, you would typically make a GET request to your employee search endpoint, including parameters for pagination (page number and size) and sorting (field and direction). For example:



This is the output response from the REST Client a paginated list of employees sorted by the specified field, along with metadata about the pagination, like the current page, total pages, and total elements. This allows you to see a portion of the data that meets the criteria, making it easier to work with large datasets.

* **Flowchart of the program:**
* **Start:** The process begins when a client sends an API request to retrieve employees with specific pagination and sorting criteria.
* **Parse Request:** The API extracts and validates pagination and sorting parameters such as page number, page size, sorting field, and sorting direction.
* S**et Defaults:** If any parameters are missing or invalid, default values are assigned (e.g., page = 0, size = 10, sortBy = "id", sortDirection = "asc").
* **Build Pageable Object:** A Pageable object is created using the parsed or default parameters.
* **Query Database:** The database is queried using the Pageable object to retrieve the corresponding page of employees.
* **Extract Data:** The retrieved employee data is extracted, focusing on the specific fields needed (e.g., employee names).
* **Format Response:** The extracted data is formatted into a response object, including metadata like total pages and current page.
* **Send Response:** The formatted response is sent back to the client.
* **End:** The process ends after the client receives the response.