Test task for a backend developer

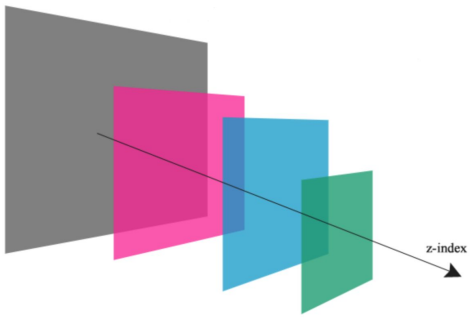
Summary

A web service to work with widgets via HTTP REST API. The service stores only widgets, assuming that all clients work with the same board.

Glossary

A Widget is an object on a plane in a Cartesian coordinate system that has coordinates (X, Y), Z-index, width, height, last modification date, and a **unique identifier**. X, Y, and Z-index are integers (may be negative). Width and height are integers > 0. Widget attributes should be not null.

A Z-index is a unique sequence common to all widgets that determines the order of widgets (regardless of their coordinates). **Gaps are allowed.** The higher the value, the higher the widget lies on the plane.



Details

Operations to be provided by the web service:

- Creating a widget. Having a set of coordinates, Z-index, width, and height, we get a complete widget description in the response. The server generates the identifier. If a Z-index is not specified, the widget moves to the foreground (becomes maximum). If the existing Z-index is specified, then the new widget shifts widget with the same (and greater if needed) upwards.
  Examples:
  1) Given - 1,2,3; New - **2**; Result - 1,**2**,3,4; Explanation: 2 and 3 has been shifted;
  2) Given - 1,5,6; New - **2**; Result - 1,**2**,5,6; Explanation: No one shifted;
  3) Given - 1,2,4; New - **2**; Result - 1,**2**,3,4; Explanation: Only 2 has been shifted;

- Changing widget data by Id. In response, we get an updated full description of the widget. We cannot change the widget id. All changes to widgets must occur atomically. That is, if we change the *XY* coordinates of the widget, then we should not get an intermediate state during concurrent reading. The rules related to the Z-index are the same as when creating a widget.

- Deleting a widget. We can delete the widget by its identifier.

- Getting a widget by Id. In response, we get a complete description of the widget.

- Getting a list of widgets. In response, we get a list of all widgets sorted by Z-index, from smallest to largest.

Requirements

- Language - Java 11; Framework - Spring Boot; Build tool - Maven;
- Do only the server-side, you don't need to do visualization;
- No need to do any authorization;
- No explicit limits on memory or runtime, but the more efficient the implementation, the better;
- You can assume that 90% of load distribution is read operations;
- The API should respect to REST architecture;
- Data should be stored in memory. You can use any classes of the standard library to organize storage. Using any external repositories and databases is forbidden.
- All changes to widgets must occur atomically. The Z-index must be unique.
- At least 30% of the code should be covered by unit and integration tests;
- Submit sources via a public git repository.

Complication

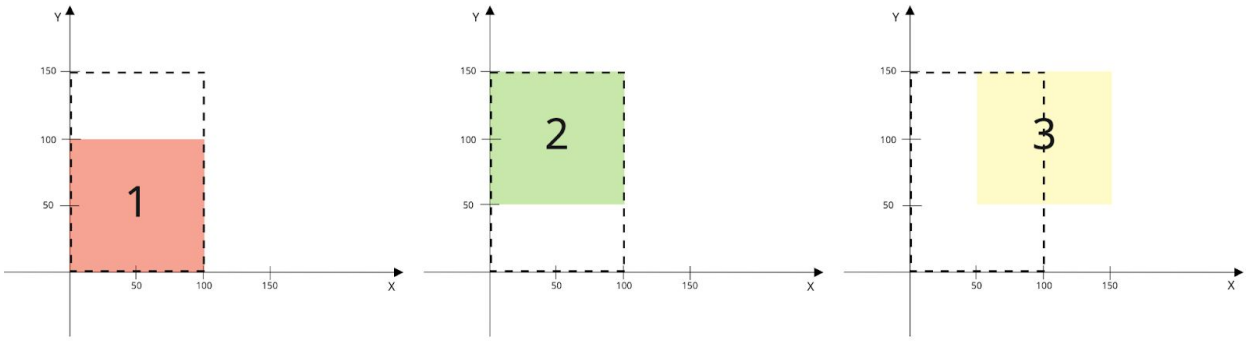
All complications are optional — you may or may not do them. You can pick any of them.

1. Pagination

Implement a pagination mechanism for a list of widgets request, which by default will produce 10 elements, but with the ability to specify a different number (up to 500). Need to modify the existing endpoint to get a list of widgets

2. Filtering

While getting the list of widgets, we can specify the area in which widgets are located. Only the widgets that fall entirely into the region fall into the result. Need to modify the existing endpoint to get a list of widgets. An important condition is to find an implementation method in which the runtime complexity of the algorithm is on average less than O(n).
For example, we have 3 widgets that have a width and a height of 100. The centers of these widgets are at points 50:50, 50:100, and 100:100. We want to get only the widgets that are inside the rectangle, the lower-left point of which is at 0:0, and the upper right is at 100:150.



In this case, we should get the first and second widgets.

3. SQL database

You need to create another storage implementation that works with SQL databases. You can use any database in-memory implementation (for example, H2).
As a result, we should get two independent implementations of the widget repository
The choice of storage implementation should be determined in the server configuration and does not require changes in the code.