IT Software Solutions for Business
EuroSkills 2025 National Competition
HUNGARY

Round 1

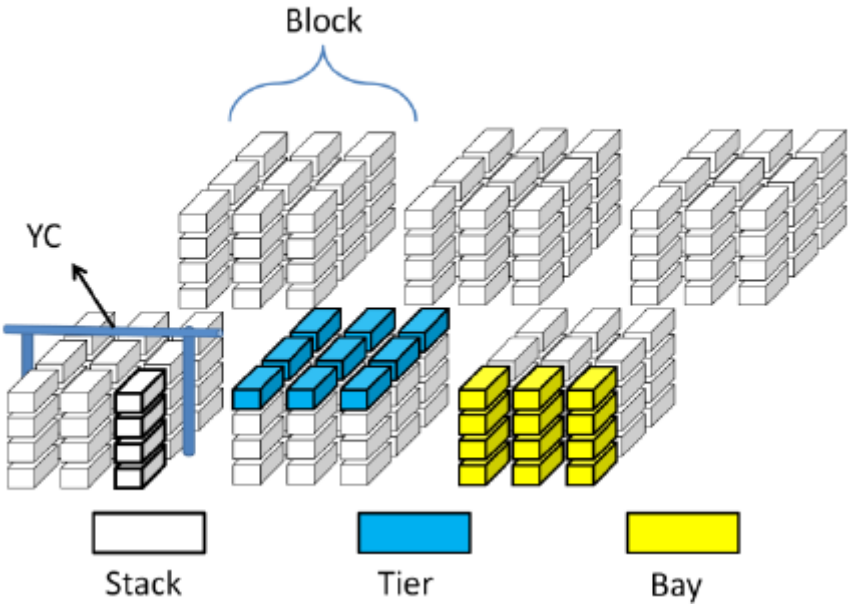
Submitted by:
Skills IT

Contents

Introduction

You used to work as a freelance software developer, but now you've applied for a job as a developer at a large software development company. The company's management wants to test your skills, so as part of the recruitment process, they asked you to develop a prototype web application for a Freight Container company. In addition to sharing up-to-date information about the yard containing the containers, it also helps operators import incoming containers and list already placed containers.

Yards are divided into blocks, each block has multiple bays, in them there are stacks and tiers.



In the prototype, all you need to develop is a dashboard and a searchable list of containers. The dashboard depicts a yard with containers, so operators can easily see the available capacity of each block, stack.. The list contains the containers and allows the operators to easily search information in them.

Each container has a block, bay, stack and tier number. In each position one and only one container can be placed. The yard has 4 blocks and each block has 5 bays, 5 stacks and 5 tiers.

Your API must use a JSON Server as its data provider.

Description of project and tasks

Your task is divided into three parts.

1. In the first part, you will develop the REST API for UIs and import functions.
2. In the second part, you have to create a web based dashboard.
3. In the third part, you will develop a searchable list for containers.

Database

1. Use the **`database.json`** from the assets folder
2. Run JSON Server (port 3000): **`json-server database.json`**

Your database has one table and has the following schema:

```
Container {
      id: string
      blockId: number
      bayNum: number
      stackNum: number
      tierNum: number
      arrivedAt: datetime (ISO-8601 YYYY-MM-DDTHH:mm:ss.sssZ)
}
```

How to submit your work

1. You have to share your work in a private GitHub repo as described in the README file of the test project GitHub repo (https://github.com/skillsit-hu/es2025-s09-hu-r1).
2. Share a link with us in email (es2025s09@skillsit.hu) from where we can download the executable files and code to test your solutions.
3. Your **README** file has to contain the instructions how to start your project backend and frontend. We prefer to give executable files or a built solution.

Part 1 – REST API (port 3001)

POST /api/containers/import

Imports a list of containers from a CSV file. (you can find an example CSV file in the assets folder).

The file pertaining the information is comma separated and has the following information:

- The container's code
- The block's number
- The bay's number
- The stack's number
- The tier's number
- The arrival time of the container in milliseconds from Epoch

If the request is successful, the endpoint returns the number of successfully inserted containers the incorrectly positioned containers' ID and the HTTP response status code must be **200**:

```
{
      "success": 35,
      "incorrectPositions": [
            "UUKJ5448686",
            "UUKJ5448612"
      ]
}
```

POST /api/containers

Imports a list of containers.

```
[
      {
            "id": "WNCZ4657336",
            "blockId": 2,
            "bayNum": 5,
            "stackNum": 3,
            "tierNum": 3,
            "arrivedAt": "2024-01-12T00:00:00.000Z"
      },
      {
            "id": "DDTI4877940",
            "blockId": 4,
            "bayNum": 4,
            "stackNum": 5,
            "tierNum": 3,
            "arrivedAt": "2024-01-12T00:00:00.000Z"
      }
]
```

If the request is successful, the endpoint returns the number of successfully inserted containers the incorrectly positioned containers' ID and the HTTP response status code must be **200**:

```
{
      "success": 35,
      "incorrectPositions": [
            "UUKJ5448686",
            "UUKJ5448612"
      ]
}
```

GET /api/blocks/stat

Retrieves statistics about each block.

The response contains the following information:

- The block id
- The capacity of the block (number of containers / full size of block, with two-decimal precision in percentage)
- The average age of the containers in the block in days (with two-decimal precision)
- The id of the oldest container in the block
- The id of the newest container in the block
- The number of empty positions in the block
- The number of fully empty bays in the block
- The number of fully empty stacks in the block

If the request is successful, the endpoint returns the statistics and the HTTP response status code must be **200**:

```
[
      {
            "blockId": 1,
            "capacity": 45.33,
            "averageAge": 12.2,
            "oldestContainerId": "DDTI4877940",
            "newestContainerId": "DDTI4877932",
            "emptyPositions": 45,
            "emptyBays": 1,
            "emptyStacks": 3
      },
      …
]
```

Part 2 – Dashboard

In this section, you need to create the Web based Dashboard for the UI to help operators visualize the yard.

The dashboard should contain the blocks and the capacity of each stack in it. You should also visualize the statistics available via the statistics endpoint.

The visualization is in your hands, there is no visual guideline to follow.

Part 3 – Container list

In this section, you need to create a list where operators can easily search containers.

The list should contain all available information for each container and should be easily accessible from the dashboard. The operator may be able to filter the data and also sort it.

The solution is in your hands, there is no visual/technical guideline to follow, you also don't have to use the already existing endpoints, you may create new ones as you see fit. Make sure all of them are properly documented.

Additional information

- Some media, icons and text have been provided for you in the media files. You are free to use these, but you can also create your own, as long as the application is still fit for purpose. **You should not use any other media files (e.g. downloaded videos, images, icons, etc.)**.
- Clean code and user interface accessibility are also important considerations.
- Do not hardcode API responses as another database will be used for testing.