

Overview:

As a packages producer I have many machines in my production plant, I want to see real time production status for my machines.

The goal of this exercise is to create an application to be able to see real time production value for my machines and delete a machine if I no longer need it.

Architecture requirements:

In this exercise you will create an application with the following recommendations :

Front	Angular (You can use regular Html and jQuery if you do not know how to use angular)
WebAPI.csproj	.Net Core Web Api
WebAPITest.csproj	Add unit tests for the web Api
Service.csproj	Business logic
Repository.csproj	Entity Framework Core And use SQL Server database

This is not an exhaustive list of projects in the solution, you can add more projects to the solution if needed.

Exercise 1: Create and initialize database and tables

1-Db name: "MachineMonitoring"

2-Table "Machine"

machineId	Int, not null, auto increment, primary key
Name	String, not null, with max length 50 chars
Description	String, nullable, with max length 250 chars

3-Table "MachineProduction":

<u>MachineProductionId</u>	Int, not null, auto increment, primary key
machineId	Int, not null, auto increment, foreign key to table Machine
totalProduction	int, not null, default value (0)

4-Data base scripts and data seed :

Generate SQL script to create data base and tables

Add to this script some data initialization (two machines with production data)

Acceptance criteria:

- Data base "MachineMonitoring" exists with the tables "**Machine**" and "**MachineProduction**"
- Scripts to Create the Database and insert data exists and runs **multiple times without errors**.
- Database access parameters stored in json configuration file

Exercise 2: Web API

Use Asp Net core Web API implement the following endpoints:

Web Method	URL	Description	Returns
GET	/machines	Get information about all machines, including their production (if any) If data exists return http status 200 If no data in response return http status 204	{[machineId: int , name : string , production : int]}
GET	/machine/{id}	Get all information about the machine with id equal to the route parameter If data exists return http status 200 If machine id does not exist return http status 404	{machineId: int , name : string, description : string}
GET	/machine/totalproduction?id= machineid	Get the value of total production for machine with id equals machineid If data exists return http status 200 If machine id does not exist return http status 404	{ totalproduction : int }
Delete	/machine/{id}	Delete a machine from the database If data exists return http status 200	0 or 1 Returns number or deleted records

Exercise 3: Web API unit test

Add unit testing to the web Api endpoints

Test all three endpoints and all http status in description

Exercise 4: Repository

Implement repository using Entity framework , database first

Exercise 5: Service

Implement business logic needed to serve the Web API

Get information about all machines, including their production



Acceptance criteria:

- The API must respect the defined input and output and http status codes
- Unit tests are passing for the Web API

Exercise 5 Front end:

- Page 1: List of available machines and the ability to navigate to the dashboard of the selected machine, when you click on the equipment 1 you go to page 2 to show details of the selected equipment

Click on the equipment to see the dashboard:

Id	Name	Production	
1	Equipment A	54	 delete
2	Equipment B	22	 delete

- Page 2: Dashboard machine name, machine description and total production (the total production is updated each **5 seconds**)

Machine	Value of machine name
Description	Value of machine description
Total production	Value of total production

Acceptance criteria:

- Delete equipment deletes from databases and refresh table content (not all the page)
- The front show list of machines and when clicking a machine, I am redirected to dashboard page
- Dashboard page gets the latest value from database each 5 seconds without refreshing the page