Inhoud

[**Introduction** 2](#_Toc73989944)

[**More in detail** 2](#_Toc73989945)

[**Payload** 2](#_Toc73989946)

[**CO2** 3](#_Toc73989947)

[**The Productionplan solution** 4](#_Toc73989948)

[**The ProductionPlan\_API project** 5](#_Toc73989949)

[**The ProductionPlan project** 5](#_Toc73989950)

[**The ProductionPlan\_Lib project** 6](#_Toc73989951)

# **Introduction**

The ***Productionplan*** tool calculates how much power each of a multitude of different powerplants need to produce (a.k.a. the production-plan) when load is given and taking into account the cost of the underlying energy sources (gas, kerosine) and the Pmin and Pmax of each powerplant.0

**More in detail**

The ***load*** is the continuous demand of power. The total load at each moment in time is forecasted. At any moment in time, all available powerplants need to generate the power to exactly match the load.   
The ***cost*** of generating power can be different for every powerplant and is dependent on external factors: The cost of producing power using a turbojet, that runs on kerosine, is higher compared to the cost of generating power using a gas-fired powerplant because of gas being cheaper compared to kerosine and because of the ***thermal efficiency*** of a gas-fired powerplant being around 50% (2 units of gas will generate 1 unit of electricity) while that of a turbojet is only around 30%. The cost of generating power using windmills however is zero. Thus deciding which powerplants to activate is dependent on the ***merit-order***.

When deciding which powerplants in the merit-order to activate (a.k.a***. unit-commitment problem***) the maximum amount of power each powerplant can produce (Pmax) obviously needs to be taken into account. Additionally gas-fired powerplants generate a certain minimum amount of power when switched on, called the Pmin.

### **Payload**

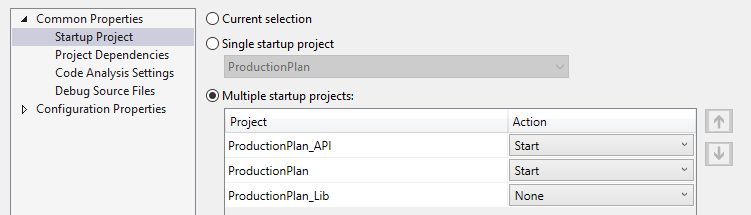
The ***payload*** contains 3 types of data:

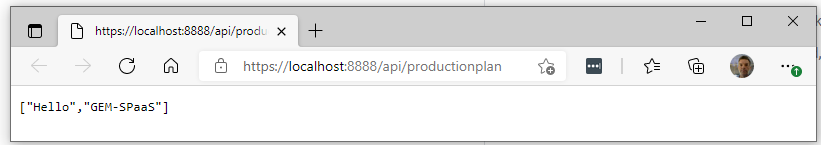
* ***load***: The load is the amount of energy (MWh) that need to be generated during one hour.
* ***fuels***: based on the cost of the fuels of each powerplant, the merit-order can be determined which is the starting point for deciding which powerplants should be switched on and how much power they will deliver. Wind-turbine are either switched-on, and in that case generate a certain amount of energy depending on the % of wind, or can be switched off.
  + ***gas(euro/MWh)***: the price of gas per MWh. Thus if gas is at 6 euro/MWh and if the efficiency of the powerplant is 50% (i.e. 2 units of gas will generate one unit of electricity), the cost of generating 1 MWh is 12 euro.
  + ***kerosine(euro/Mwh)***: the price of kerosine per MWh.
  + ***co2(euro/ton):*** the price of emission allowances (optionally to be taken into account).
  + ***wind(%)***: percentage of wind. Example: if there is on average 25% wind during an hour, a wind-turbine with a Pmax of 4 MW will generate 1MWh of energy.
* ***powerplants***: describes the powerplants at disposal to generate the demanded load. For each powerplant. is specified:
  + **name**:
  + ***type***: gasfired, turbojet or windturbine.
  + ***efficiency***: the efficiency at which they convert a MWh of fuel into a MWh of electrical energy. Wind-turbines do not consume 'fuel' and thus are considered to generate power at zero price.
  + ***pmax***: the maximum amount of power the powerplant can generate.
  + ***pmin***: the minimum amount of power the powerplant generates when switched on.

**CO2**Taken into account that a gas-fired powerplant also emits CO2, the cost of running the powerplant should also take into account the cost of the emission allowances. For this challenge, you may take into account that each MWh generated creates 0.3 ton of CO2.

# **The Productionplan solution**

The productionPlan solution is created in ***Microsoft Visual Studio Community 2019,***Microsoft .NET Framework Version 4.8.04084.  
  
**Startup**

The solution is configured with multiple startup projects. The ***ProductionPlan\_API*** project starts first, the ***ProductionPlan*** project (UI) is second.  


The Web API ***ProductionPlan\_API*** is started in a separated browser window.  


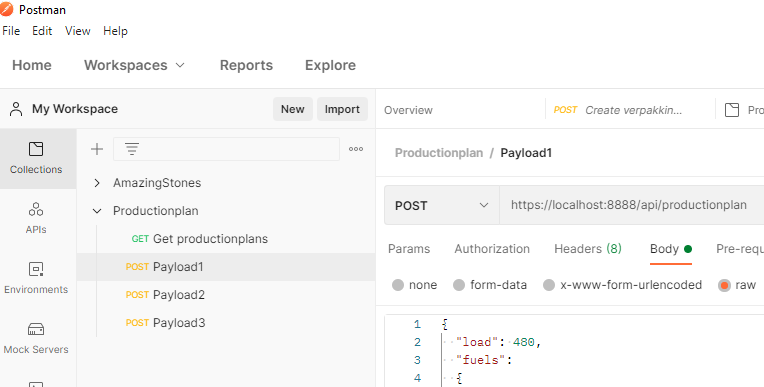
And the application is also started in a separated browser window.

**The ProductionPlan\_API project**

The API starts with the ProductionPlan\_API profile defined in the ***launchSettings.json.*** The applicationUrl = "https://localhost:8888;http://localhost:8889".  
 **Handling requests**  
The ProductionPlanController class handles the requests.   
The Post action ([HttpPost]) handles the POST request in our example.  
The ProductionPlanManager class calculates the productionplan based on the payload.  
It calculates how much power each of a multitude of different powerplants need to produce (a.k.a. the production-plan).

* Order powerplants by merit-order. Those with the lowest costs are the first ones to be brought online to meet demand.
* Taken into account that a gas-fired and a turbojet powerplant emits CO2.
* The sum of the power generated by each of the different powerplants is exactly equal to the load specified in the payload.

**Handling exceptions**  
The ProductionPlanController class uses the ***Exception Handling Middleware*** to produce a consistent error payload format in the response body.  
In the startup.cs - public void Configure() attribute routing is applied to the ProductionPlanController actions (ErrorLocalDevelopment and Error).

**Testing with Postman**  
The Web API is tested with the tool Postman. In the workspace the collection “Productionplan” is created with 4 requests.  
  


**The ProductionPlan project**

This project is a WPF core project. The application provides the GUI for the calculation of the productionplans.  
  
**Simulations 24 hour**  
In the MainWindow on the tab “Simulations 24 hours” it is possible to run 6 different   
***“24*** ***hour*** ***simulation”***. Each simulation can be started with its corresponding button. When the application starts it runs the simulation “Normal load + Normal”.  
For each hour in the day a ***productionplan*** is calculated. Based on the load and the windpercentage, the productionplan decides which powerplants should be switched on and how much power they will deliver in that specific hour.

* There are 3 “24 hour simulation”with a ***normal load*** during the dayand3 “24 hour simulations” with ***a heavy load*** during the day.  
  The ***load*** is indicated with a color: green (0-500 MWh), orange (500 -1000 MWh) and red (1000-1500 MWh). During a day the load will be fluctuate:   
  at 05 and 21 hour the demand from industry will be bigger.
* The ***windpercentage*** is also indicated with a color: light blue (0-33 %),   
  dodger blue (33 -66 %) and blue (66-100 %). During a day the wind percentage will be fluctuate. Each group of 3 “24 hour simulation”(normal- and heavy load load) have a “Normal”, “No wind” and “storm” windpercentage.

**Payloads**In the MainWindow on the tab “Payloads” it is possible to run the 3 example payloads. Each payload can be started with its corresponding button. When switched to this tab the productionplan for the “Payload1” is calculated.

**Calling the productionplan Web API**The ProductionPlanController class mocks all the data for the simulations and payloads (public void FillSimulationData(enumSimulationType simulationType)).  
For each hour in the day a productionplan is calculated. Based on the load and the windpercentage, the productionplan decides which powerplants should be switched on and how much power they will deliver in that specific hour.  
The public async Task<Lib.ProductionPlan> Simulate(double load, double wind) is the method that calls the Web API. The model class Payload is serialized to JSON to create a HTTP-request. The JSON in the HTTP-response is deserialized to the model class ProductionPlan.

**The ProductionPlan\_Lib project**

This project is a Core library project. The model classes are shared with the ProductionPlan and ProductionPlan\_API projects.