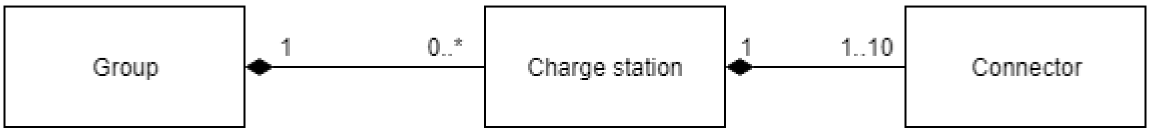
GreenFlux Smart Charging Assignment

The assignment is very simplified version of real requirements for capacity control in Smart Charging.

Domain model:



Charge station – has a unique identifier (cannot be changed), name (can be changed), multiple connectors (at least one, but not more than 5).

Connector – has numerical identifier unique per charge station with (possible range of values from 1 to 5), Max current in Amps (can be changed) – value greater than zero.

Group – has a unique identifier (cannot be changed), name(can be changed), *capacity in Amps* (can be changed) – value greater than zero. Group can contain multiple charge stations.

Functional requirements:

1. A connector cannot exist in the domain without a charge station.
2. Max current of an existing connector can be changed.
3. The charge station cannot exist in the domain without *Group*. The charge station can be only in one group at the same time.
4. Only one charge station can be added/removed from a group in one call.
5. Group can be created, update, and removed.
6. If a group is removed, all charge stations in the group should be removed as well.
7. The capacity of a group should always be great or equal to the sum of Max current in Amps of the connector of all charge stations in the group.
8. Based on the previous paragraph, if the capacity of a group is not enough when adding a new connector – API should return a response with a suggestion of which connectors of which charge stations should be removed from the group to free space for the new connector.
    a. The suggestion should be made in the most optimal way – should be removed the minimum connectors as possible to free the exact amount of capacity for the new value.
    b. If there are multiple equally optimal ways, the endpoint should return multiple suggestions.

Technical requirements:

1. Create RESTful ASP.NET Core API according to described requirements.
2. Use any convenient database to store data. It can be in-memory.
3. Think about the performance of the solution.
4. Cover your code by necessary in your opinion unit and/or integration tests.
5. Use a local Git repository for your code.
6. Nice to have Swagger.
7. Provide a ready-to-run solution (Visual Studio or Visual Studio Code) via GitHub or any other public Git repository.

Non-technical requirement:

1. Create a readme.md file with how much time or percentage of time you spent on each task of the assignment. For example, 20% creating the visual studio solution with all references and architectural layers; 10% understanding the requirement; 10% designing the solution; 40% coding the solution; 5% creating the source repository and doing the right source control; and 15% testing the solution.
2. In the same readme.md file, also explain what you would improve in your assignment (code solution) if you had more time.

We will evaluate if your code

1. Solve the problem in the most efficient way.
2. Has no bugs.
3. Is clean (not over-engineered).
4. Testable (happy and unhappy flows are covered by tests).
5. Is maintainable (what if we need to extend your solution in the future).

Please don't spend time on components that are not required to solve the problem. There are no requirements to use CQRS or aggregation roots or Event Sourcing but you can use any patterns if it helps you to solve the problem most efficient way.

Remember – you should be ready to argue your decisions.

The result doesn't need to be fully finished, as long as it shows your skills as a .NET developer. Please send us back the source code. Preferably via a GitHub URL repository, with a ready-to-compile solution.

We appreciate and expect questions!

Your additional questions can be sent by emails:

Matheus Calache: matheus.calache@greenflux.com

Raman Tsitou: raman.tsitou@greenflux.com

The subject of your email should start with **[assignment]**.