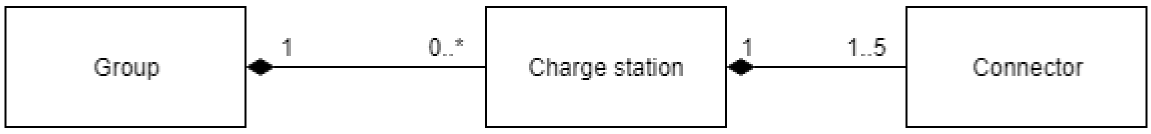
GreenFlux Smart Charging Assignment

Version: 9

Please develop a solution given the following model and requirements.

Domain model:



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.

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.

Functional requirements:

1. *Group*/*Charge Station*/*Connector* can be created, updated and removed.
2. If a *Group* is removed, all *Charge Stations* in the *Group* should be removed as well.
3. Only one *Charge Station* can be added/removed to a *Group* in one call.
4. The *Charge Station* can be only in one *Group* at the same time.
   The *Charge Station* cannot exist in the domain without *Group*.
5. A *Connector* cannot exist in the domain without a *Charge Station*.
6. The Max current in Amps of an existing *Connector* can be changed (updated).
7. The capacity in Amps of a *Group* should always be great or equal to the sum of the Max current in Amps of the *Connector* of all *Charge Stations* in the *Group*.
8. If the capacity in Amps of a *Group* is not enough when adding a new *Connector*, the API should reject the new *Connector*.

Technical requirements:

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

Non-technical requirement:

Create a *readme.md* file. If your assignment requires some extra operations to run it (create a database for example) detailed instruction must be added.

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. You can use any patterns if it helps you to solve the problem most efficient way. And remember, you should be ready to argue your decisions.

Please send us back the source code (zip file) with a ready-to-compile solution. Preferably, a clean solution without the bin and obj folders, and without other ignored files (like if you were using the *.gitignore* file for Visual Studio repositories).

We appreciate and expect questions!

The result of your assignment must be sent to Rob and Matheus via e-mail.

- Rob van Elburg: rob.van.elburg@greenflux.com
- Matheus Calache: matheus.calache@greenflux.com

Your additional questions can be sent to Matheus and Raman via e-mail:

- Matheus Calache: matheus.calache@greenflux.com
- Raman Tsitou: raman.tsitou@greenflux.com

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