The solution was coded in Visual Studio 2019 Professional. Version 16.9.6, using Web API/ ASP NET Core 3.1

For the sake of simplicity, I have used Entity Framework Core as an In-Memory database for ASP NET Core, and Code First approach for creating the database objects and sample data for testing.

The following test data is added automatically when application starts for testing purposes:

Gateways:

Gateway {Id= "e0c653d1-248b-4877-b629-e9030dce6095", Name= "Gateway-1", IpAddress= "127.0.0.1"}

Gateway {Id= "138c4789-ad19-47e2-9b84-8375caf0a792", Name= "Gateway-2", IpAddress= "192.168.0.1"}

Gateway {Id= "9dbae714-b2e7-4d46-89cb-81ce52a28741", Name= "Gateway-3", IpAddress= "192.168.0.254"}

Peripheral Devices:

PeripheralDevice

{

Uid = 10000,

Vendor = "Siemens",

DateCreated = new DateTime(2000, 10, 01),

Online = true,

GatewayID = "e0c653d1-248b-4877-b629-e9030dce6095"

},

PeripheralDevice

{

Uid = 10001,

Vendor = "Sony",

DateCreated = new DateTime(2010, 01, 01),

Online = true,

GatewayID = "e0c653d1-248b-4877-b629-e9030dce6095"

},

PeripheralDevice

{

Uid = 10002,

Vendor = "Sanyo",

DateCreated = new DateTime(2018, 02, 03),

Online = false,

GatewayID = "138c4789-ad19-47e2-9b84-8375caf0a792",

}

How to test the API:

1. Operation for storing a Gateway

**POST**: <http://localhost:5000/api/gateways>

Request body (JSON)

{

"name": "Gateway Name",

"ipaddress":"200.1.25.56"

}

1. Operation for displaying information about all stored gateways (and their devices)

**GET**: <http://localhost:5000/api/gateways>

1. Operation for displaying details for a single gateway (and its devices)

**GET**: <http://localhost:5000/api/gateways/1>

1. Operation for adding a peripheral to a gateway

**POST**: <http://localhost:5000/api/devices>

Request body (JSON)

{

"vendor": "Sony",

"datecreated":"2019-01-01",

"online": true,

"gatewayid": "1"

}

1. Operation for removing a peripheral

DELETE: <http://localhost:5000/api/devices/10002>

* When storing a gateway to the database, we only have to specify the “name” and “ipaddress” attributes as the gateway ID is being generated automatically (defined at DBContext)
* Notice that the action of adding a peripheral to a gateway takes just the device details including the gateway ID to which is related. As the in-memory database is not a relational database, it does not enforce the foreign key constraints, so, in this particular case, no errors will be raised when adding a peripheral to an inexistent gateway. If we use an actual relational database engine, then and error would be returned. There’s no need to specify the peripheral Uid as it is generated automatically.
* As there’s no specification, I’m assuming that when a peripheral is not online, it is still managed by the corresponding gateway, so, the action of removing it from a gateway would be just deleting it. As long as the peripheral device UID is unique, we need to provide just the peripheral UID as parameter for removing a peripheral from a gateway.
* Attached a Postman collection for test calls.