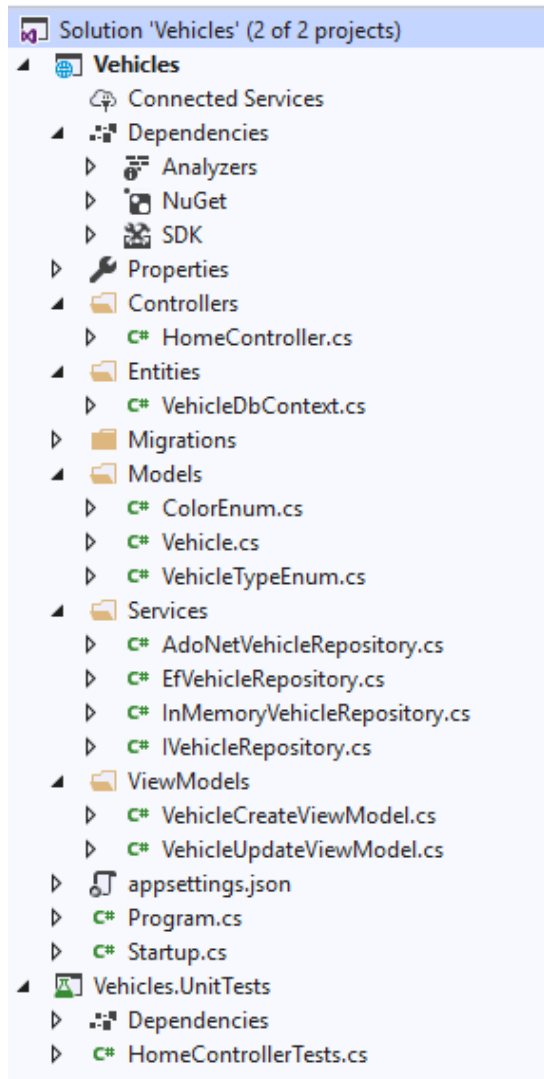


Opdracht Programmeren 3

We bouwen een API voor het beheren van verschillende types van voertuigen. Bouw in Visual Studio 2019 een solution met onderstaande structuur.



De Models folder bevat onderstaande klasse en enums:

```
public class Vehicle
{
    public int Id { get; set; }
    public string Make { get; set; }
    public string Model { get; set; }
    public string VIN { get; set; }
    public VehicleTypeEnum Type { get; set; }
    public ColorEnum Color { get; set; }
}
```

```
public enum VehicleTypeEnum
{
    Truck,
    Bus,
    Car,
    Suv
}
```

```
public enum ColorEnum
{
    Red,
    Green,
    Blue
}
```

De IVehicleRepository onder services voorziet onderstaande methods:

```
public interface IVehicleRepository
{
    2 references | 0 exceptions
    IEnumerable<Vehicle> GetAll();
    5 references | 0 exceptions
    Vehicle Get(int id);
    2 references | 0 exceptions
    void Add(Vehicle vehicle);
    3 references | 0/1 passing | 0 exceptions
    void Delete(Vehicle vehicle);
    2 references | 0 exceptions
    void Update(Vehicle vehicle);
}
```

Als je de applicatie opstart krijg je onderstaande swagger te zien:

The screenshot displays the Swagger UI for an application. It features a 'Home' section with five API endpoints and a 'Models' section with two data models.

Home

- GET** /Home
- PUT** /Home
- POST** /Home
- DELETE** /Home
- GET** /Home/id

Models

VehicleCreateViewModel {

- make* string
- model* string
- vin* string
- type integer(\$int32)
- Enum: > Array [4]
- color integer(\$int32)
- Enum: > Array [3]

}

VehicleUpdateViewModel {

- vin* string
- color integer(\$int32)
- Enum: > Array [3]

}

Via deze swagger kan je voertuigen toevoegen. Zorg ervoor dat Make, Model en VIN verplicht op te geven zijn:

POST /Make

Parameters

Cancel

| Name | Description |
|----------------------------------|---|
| vehicleCreateViewModel (body) | <div>Edit Value Model</div> <div><pre>{ "make": "BMW", "model": "X5", "vin": "1-400-000", "type": 1, "color": 2 }</pre></div> <div>Cancel</div> <div>Parameter content type application/json-patch+json</div> |

Execute

Clear

Responses

Response content type
application/json

Call

```
curl -X POST "http://localhost:8080/Make" -H "accept: application/json" -H "Content-type: application/json-patch+json" -d '{ "make": "BMW", "model": "X5", "vin": "1-400-000", "type": 1, "color": 2 }'
```

Request URL

http://localhost:8080/Make

Server response

Code

Details

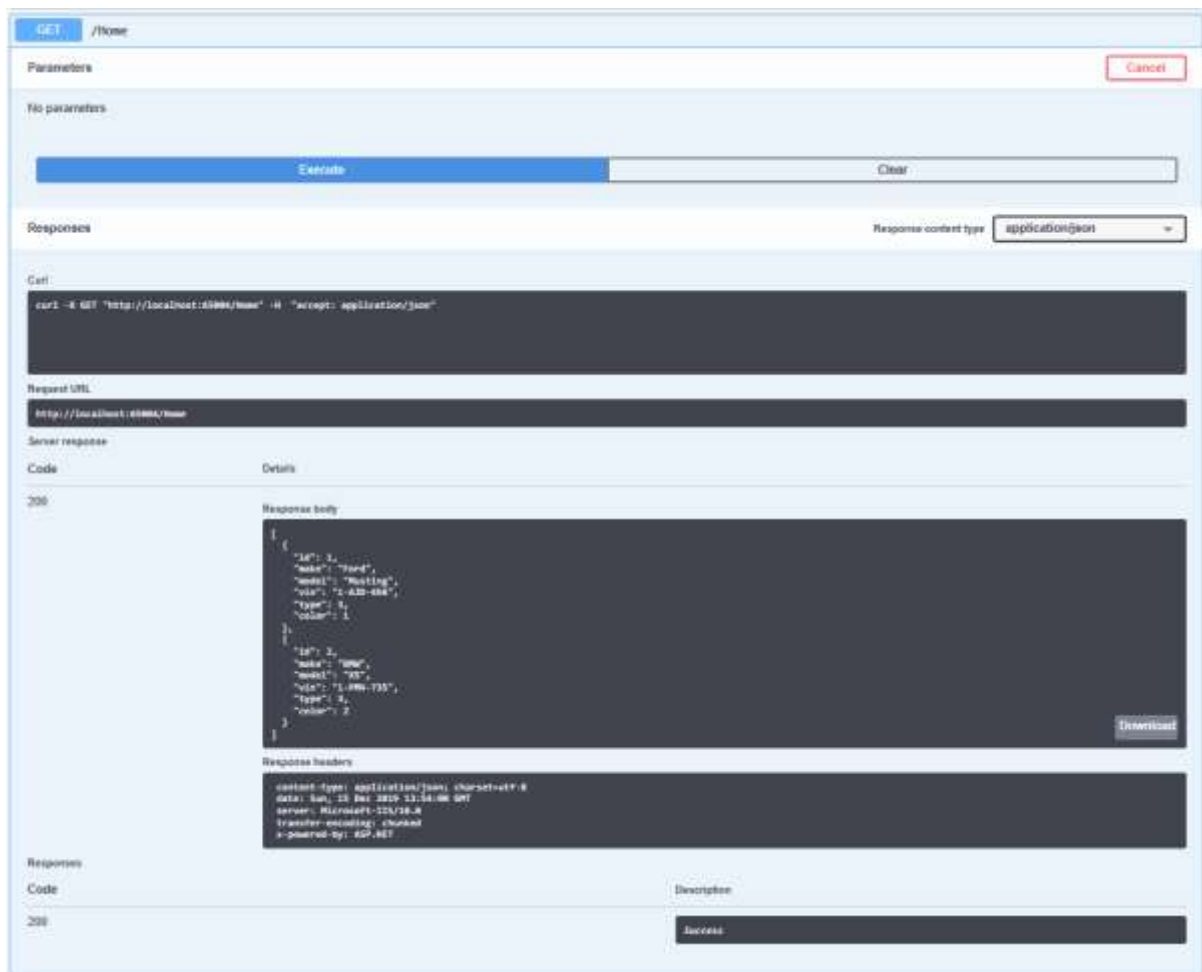
201
(application/json)

Response body

```
{
  "id": 2,
  "make": "BMW",
  "model": "X5",
  "vin": null,
  "type": 1,
  "color": 2
}
```

Download

Verder kan je via de swagger al de voertuigen opvragen die in het systeem zitten:



The screenshot shows the Swagger UI for the `GET /home` endpoint. The `Parameters` section is empty. The `Responses` section shows a `200` response with a `Content-Type` of `application/json`. The `Response body` is a JSON array of two vehicle objects. The `Request URL` is `http://localhost:8080/home`. The `Server response` section shows the `Code` as `200` and the `Details` as `Success`.

Parameters

No parameters

Responses

Response content type: `application/json`

Cell

```
curl -X GET "http://localhost:8080/home" -H "accept: application/json"
```

Request URL

```
http://localhost:8080/home
```

Server response

Code

200

Details

Response body

```
{
  "id": 1,
  "make": "Ford",
  "model": "Mustang",
  "year": "1965-66",
  "type": 1,
  "color": 1
},
{
  "id": 2,
  "make": "BMW",
  "model": "X5",
  "year": "1999-2000",
  "type": 2,
  "color": 2
}
```

Response headers

```
Content-Type: application/json; charset=utf-8
Date: Sun, 23 Sep 2019 13:54:06 GMT
Server: Microsoft-IIS/10.0
Transfer-Encoding: chunked
x-powered-by: ASP.NET
```

Responses

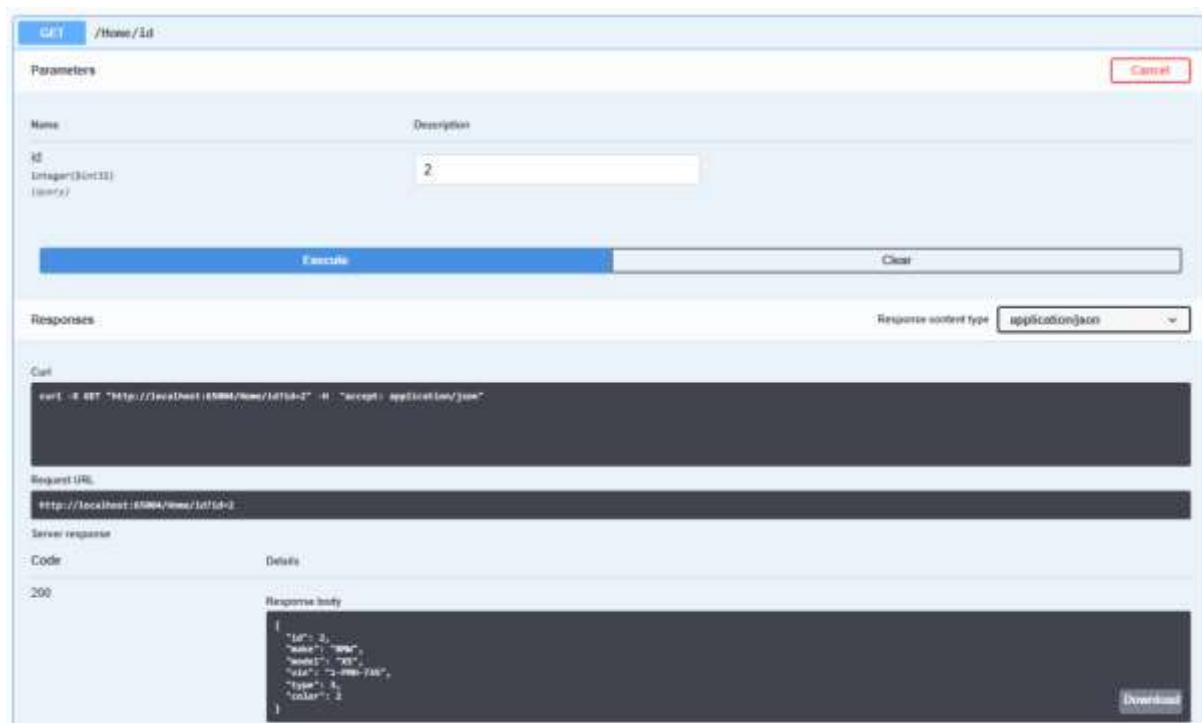
Code

200

Description

Success

Je kan ook de details opvragen van een specifiek voertuig op basis van zijn id:



The screenshot shows the Swagger UI for the `GET /home/{id}` endpoint. The `Parameters` section shows a `id` parameter of type `Integer(32-bit)` with a value of `2`. The `Responses` section shows a `200` response with a `Content-Type` of `application/json`. The `Request URL` is `http://localhost:8080/home/{id=2}`. The `Server response` section shows the `Code` as `200` and the `Details` as `Success`.

Parameters

Name

Description

id

Integer(32-bit)

(nullable)

2

Responses

Response content type: `application/json`

Cell

```
curl -X GET "http://localhost:8080/home/{id=2}" -H "accept: application/json"
```

Request URL

```
http://localhost:8080/home/{id=2}
```

Server response

Code

200

Details

Response body

```
{
  "id": 2,
  "make": "BMW",
  "model": "X5",
  "year": "1999-2000",
  "type": 2,
  "color": 2
}
```

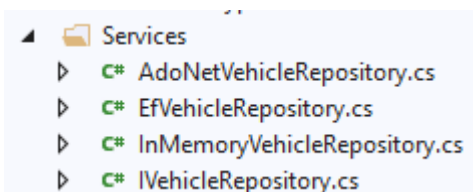
Verder kan je color en VIN aanpassen voor een bepaalde id:

The screenshot shows a REST client interface for a PUT request to the /Home endpoint. The 'Parameters' section lists two parameters: 'id' (Integer(32x132), query) with a value of '2', and 'vehicleUpdateViewModel' (body). The 'vehicleUpdateViewModel' is expanded to show a JSON body: { "vin": "1-ABC-123", "color": 4 }. There are 'Cancel' buttons for both the parameters and the body editor.

En je kan ook nog items verwijderen op basis van id:

The screenshot shows a REST client interface for a DELETE request to the /Home endpoint. The 'Parameters' section lists one parameter: 'id' (Integer(32x132), query) with a value of '2'. Below the parameters is a large blue 'Execute' button. The 'Responses' section shows a dropdown for 'Response content type' set to 'application/json'. The bottom section is labeled 'Code' and 'Description'.

Je maakt 3 implementaties van IVehicleRepository:



Verder voeg je nog een unit test project toe. Hier test je de delete method op de home controller. Je test of de controller correct omgaat met het proberen verwijderen van items die niet bestaan. Dit doe je door de IVehicleRepository uit te mocken met Moq.

Stuur je opdracht ten laatste in op 19/01/2020 via digitap.