using Microsoft.AspNetCore.Mvc;

namespace AKWebAPI.Controllers

{

[Route("[controller]")]

[ApiController]

public class PrimitiividController : ControllerBase

{

// GET: primitiivid/hello-world

[HttpGet("hello-world")]

public string HelloWorld()

{

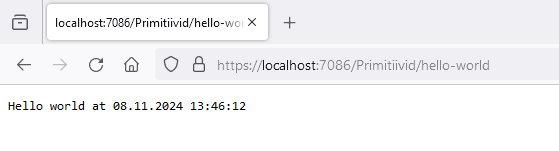
return "Hello world at " + DateTime.Now;

}

}

}

**Suuname kontrollerinimele ja metoodile, ning näeme sisu**



[Route("[controller]")]

[ApiController]

public class PrimitiividController : ControllerBase

{

// GET: primitiivid/hello-world

[HttpGet("hello-world")]

public string HelloWorld()

{

return "Hello world at " + DateTime.Now;

}

// GET: primitiivid/hello-variable/mari

[HttpGet("hello-variable/{nimi}")]

public string HelloVariable(string nimi)

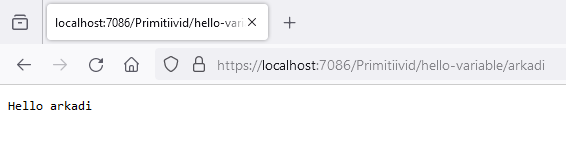
{

return "Hello " + nimi;

}

}

**Pärast metoodi nimet anname tähenduse /oma nimi**



// GET: primitiivid/do-logs/5

[HttpGet("do-logs/{arv}")]

public void DoLogs(int arv)

{

for (int i = 0; i < arv; i++)

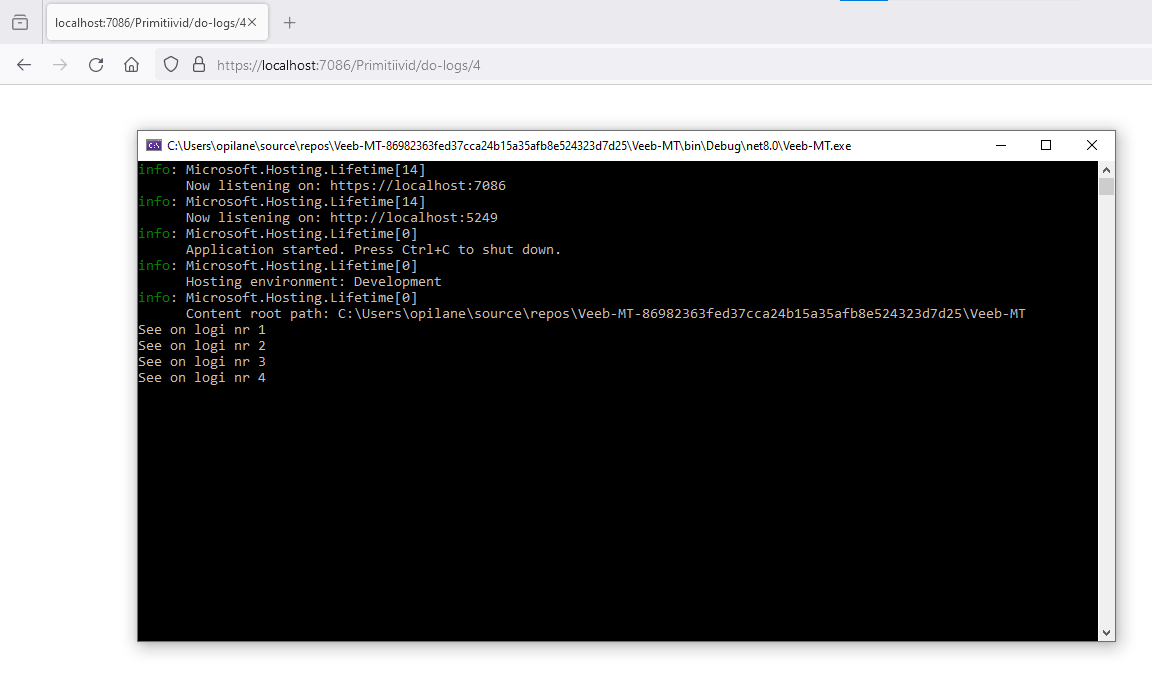
{

Console.WriteLine("See on logi nr " + i);

}

}

**Loob käsureal logid antud arvuga pärast metoodit**



// GET: primitiivid/aasta/2000

[HttpGet("aasta/{nr1}")]

public string Synniaasta(string nr1)

{

string AA1970 = "Sina oled vana";

string AA2000 = "Sina oled noor";

string AA2010 = "Sina oled väga noor";

string NoData = "Palun, sisesta arv";

int numericvalue;

bool isNumber = int.TryParse(nr1, out numericvalue);

if (!isNumber)

{

return NoData;

}

if (numericvalue < 1970)

{

return AA1970;

}

else if (numericvalue < 2000)

{

return AA2000;

}

else if (numericvalue < 2010)

{

return AA2010;

}

else

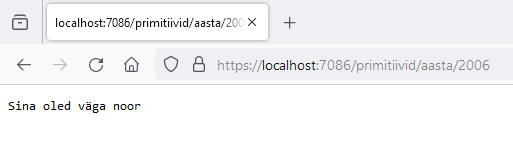
{

return NoData;

}

}

**Sisestame oma sündimisaasta ja näeme järgmise tulemuse**



// GET: primitiivid/paar-arvud

[HttpGet("paar-arvud")]

public string Paararvud()

{

Random rand = new Random();

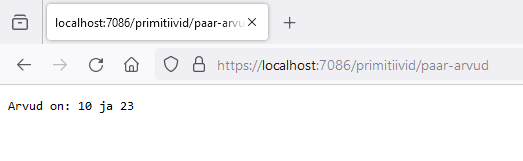
int Nr1 = rand.Next(1,100);

int Nr2 = rand.Next(1,100);

return $"Arvud on: {Nr1} ja {Nr2}";

}

**Metood genereerib paar arvudele 1-100 väärtuse**



[Route("[controller]")]

[ApiController]

public class ToodeController : ControllerBase

{

private static Toode \_toode = new Toode(1, "Koola", 1.5, true);

// GET: toode

[HttpGet]

public Toode GetToode()

{

return \_toode;

}

// GET: toode/suurenda-hinda

[HttpGet("suurenda-hinda")]

public Toode SuurendaHinda()

{

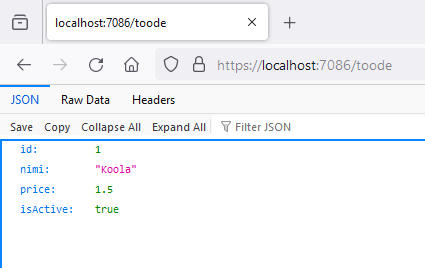
\_toode.Price = \_toode.Price + 1;

return \_toode;

}

}

**Metood suurendab hinne ühele**



// GET: toode/suurenda-hinda

[HttpGet("suurenda-hinda")]

public Toode SuurendaHinda()

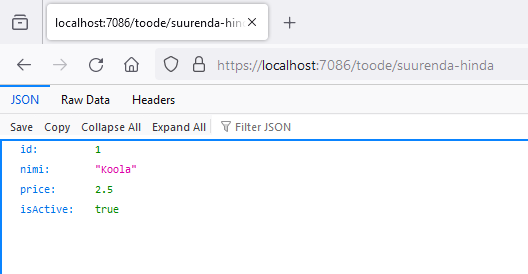
{

\_toode.Price = \_toode.Price + 1;

return \_toode;

}

**suurendab hinda 1 võrra**



// GET: toode/muuda-aktiivsust

[HttpGet("muuda-aktiivsust")]

public Toode MuudaAktiivsust()

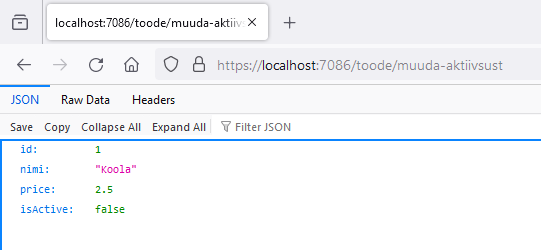
{

\_toode.IsActive = !\_toode.IsActive;

return \_toode;

}

**Muudab aktiivsust true <-> false**



// GET: toode/muuda-nime/Pepsi

[HttpGet("muuda-nime/{uusNimi}")]

public Toode MuudaNime(string uusNimi)

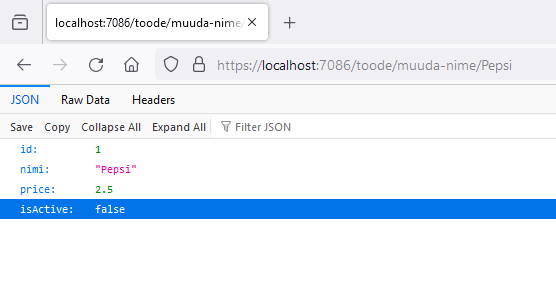
{

\_toode.Nimi = uusNimi;

return \_toode;

}

**Määrab uue nime**



// GET: toode/muuda-hinda/3

[HttpGet("muuda-hinda/{kordaja}")]

public Toode MuudaHinda(double kordaja)

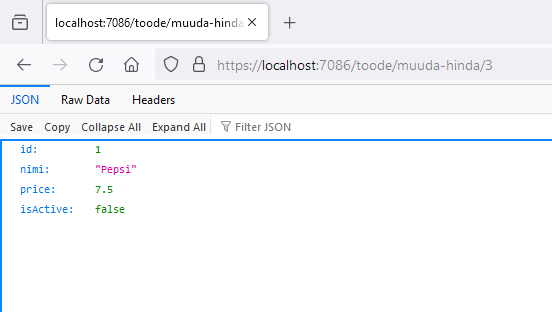
{

\_toode.Price = \_toode.Price \* kordaja;

return \_toode;

}

**Korrutab hinna kordajaga läbi**



[HttpGet("kustuta/{index}")]

public List<Toode> Delete(int index)

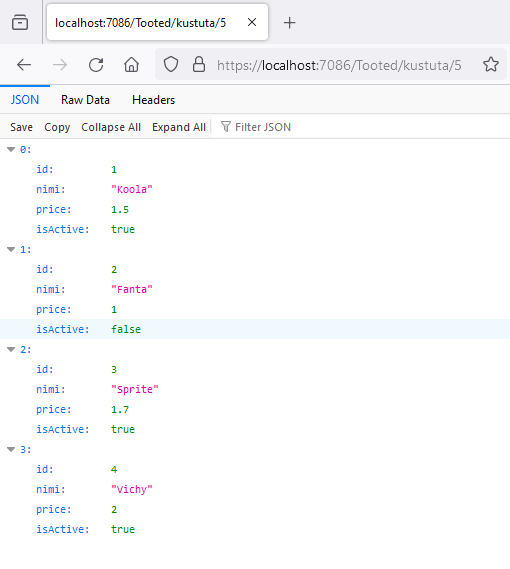
{

\_tooted.RemoveAt(index - 1);

return \_tooted;

}

**Kustutab 1 toode**



[HttpPut("uuenda/{id}/{nimi}/{hind}/{aktiivne}")]

public List<Toode> Update(int id, string nimi, double hind, bool aktiivne)

{

var existingToode = \_tooted.FirstOrDefault(t => t.Id == id);

if (existingToode != null)

{

existingToode.Nimi = nimi;

existingToode.Price = hind;

existingToode.IsActive = aktiivne;

}

else

{

Toode toode = new Toode(id, nimi, hind, aktiivne);

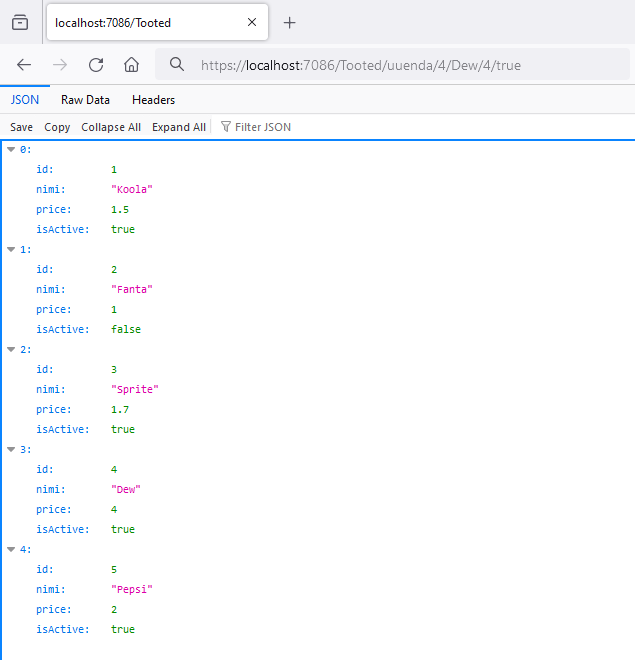
\_tooted.Add(toode);

}

return \_tooted;

}

**Lisab uue toote või uuendab olemasolevat toodet**



[HttpGet("hind-dollaritesse/{kurss}")]

public List<Toode> Dollaritesse(double kurss)

{

for (int i = 0; i < \_tooted.Count; i++)

{

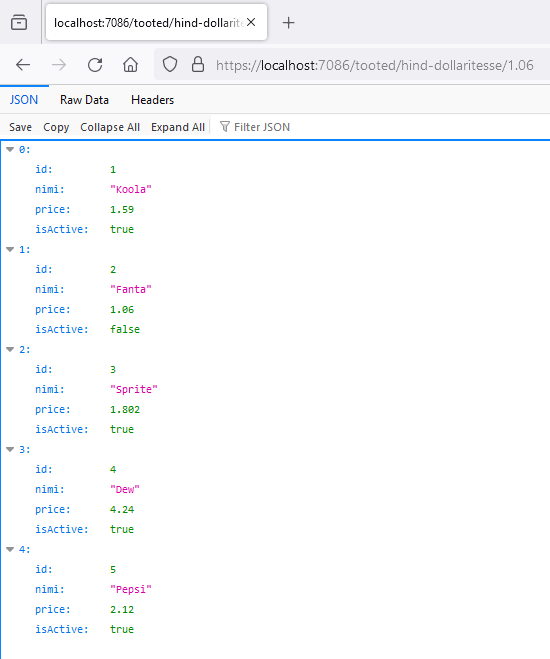
\_tooted[i].Price = \_tooted[i].Price \* kurss;

}

return \_tooted;

}

**Muudab hind korrutada määratud kurssile**



[HttpGet("muuda-aktiivsus-valeks")]

public List<Toode> DeactivateAll()

{

foreach (var t in \_tooted)

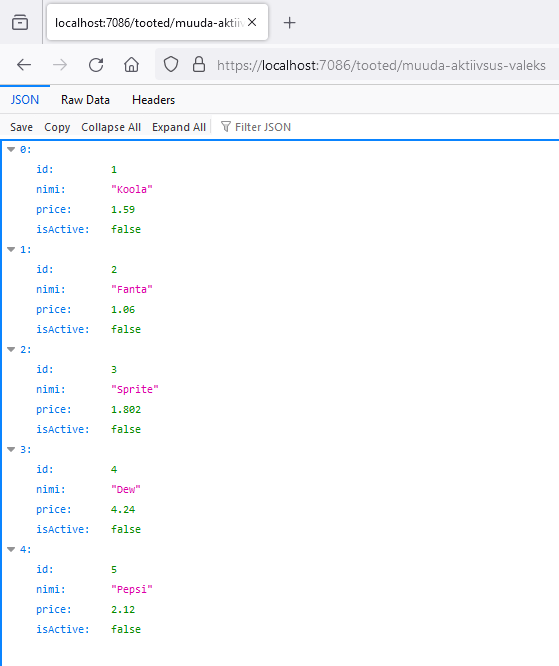
{

t.IsActive = false;

}

return \_tooted;

}

**Muudab kõik tegevused valeks** 

[HttpGet("korgeim-hind")]

public ActionResult<Toode> GetMostExpensiveToode()

{

if (\_tooted.Count == 0)

{

return NotFound("Tooteid pole saadaval.");

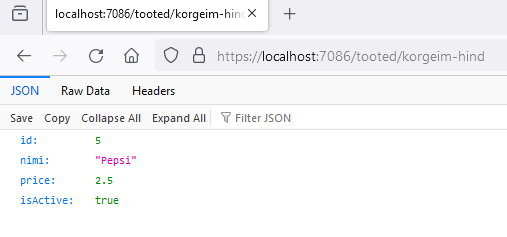
}

var kallimToode = \_tooted.OrderByDescending(t => t.Price).FirstOrDefault();

return kallimToode;

}

**Näitab kõige kallimat toodet**



[HttpGet("{index}")]

public ActionResult<Toode> GetToodeByIndex(int index)

{

if (index < 0 || index >= \_tooted.Count)

{

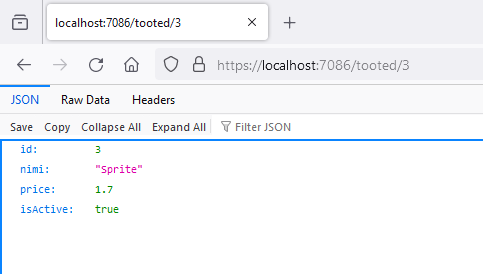
return NotFound("Toodet ei leitud.");

}

return \_tooted[index - 1];

}

**Näitab konkreetse indeksiga toodet**



[HttpGet("kustuta-koik")]

public List<Toode> DeleteAll()

{

\_tooted.Clear();

return \_tooted;

}

**Kustutab kõik**

