using Microsoft.AspNetCore.Mvc;

namespace veeb.Controllers

{

[Route("[controller]")]

[ApiController]

public class PrimitiividController : ControllerBase

{

// GET: primitiivid/hello-world

[HttpGet("hello-world")]

public string HelloWorld()

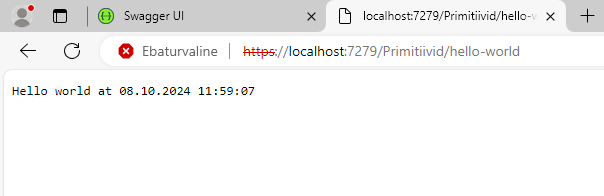
{

return "Hello world at " + DateTime.Now;

}

}

}



Joonis . 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}")]

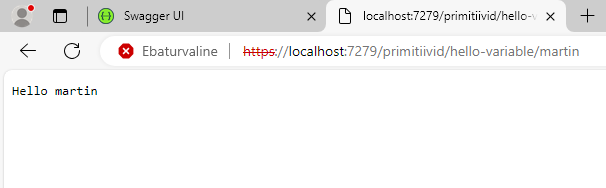
public string HelloVariable(string nimi)

{

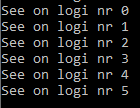
return "Hello " + nimi;

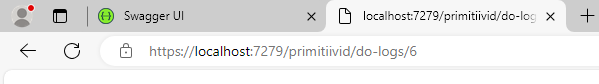
}

}



Joonis . Pärast metoodi nimet anname tähenduse /oma nimi





Joonis . Loob käsureal logid antud arvuga pärast metoodit

// 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);

}

}

// 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;

}

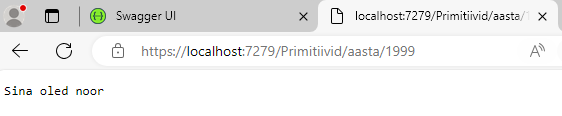
else

{

return NoData;

}

}



Joonis . Sisestame oma sündimisaasta ja näeme järgmise tulemuse

// GET: primitiivid/paar-arvud

[HttpGet("paar-arvud")]

public string Paararvud()

{

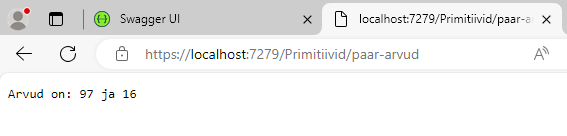
Random rand = new Random();

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

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

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

}



Joonis . 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()

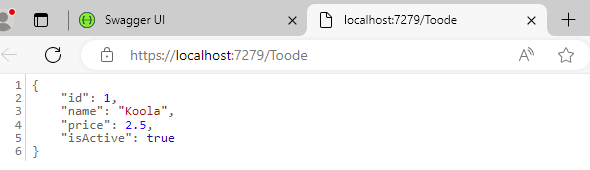
{

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

return \_toode;

}

}



Joonis . Metood suurendab hinne ühele

// GET: toode/false-true-false

[HttpGet("false-true-false")]

public Toode FalseTrueFalse()

{

if(\_toode.IsActive == true)

{

\_toode.IsActive = \_toode.IsActive = false;

}

else

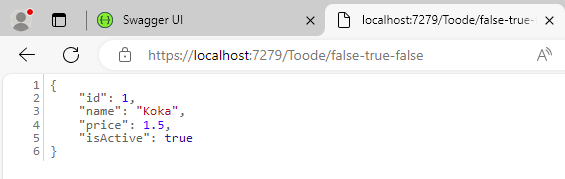
{

\_toode.IsActive = \_toode.IsActive = true;

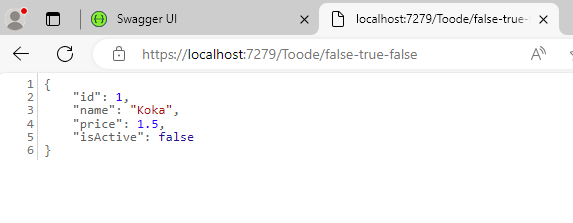
}

return \_toode;

}



Joonis . Näeme, et algusel on isActive true



Joonis . Pärast metoodi käivitamist näeme, et isActive muutub false -le

// GET: toode/ümbernimetamine

[HttpGet("ümbernimetamine")]

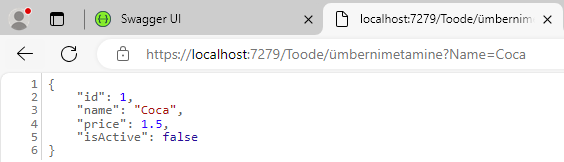
public Toode FalseTrueFalse(string Name)

{

\_toode.Name = Name;

return \_toode;

}



Joonis . Toode nimi oli Koka, muudame selle Coca -ks

// GET: toode/uushind

[HttpGet("uushind")]

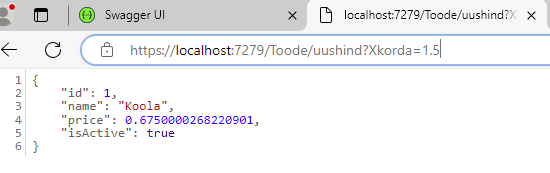
public Toode Uushind(float Xkorda)

{

\_toode.Price \*= Xkorda;

return \_toode;

}



Joonis . Muudame hind väärtusele korrutada 1.5 hind

// https://localhost:port/tooted

[HttpGet]

public List<Toode> Get()

{

return \_tooted;

}



Joonis . Näitab kõik tooded

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

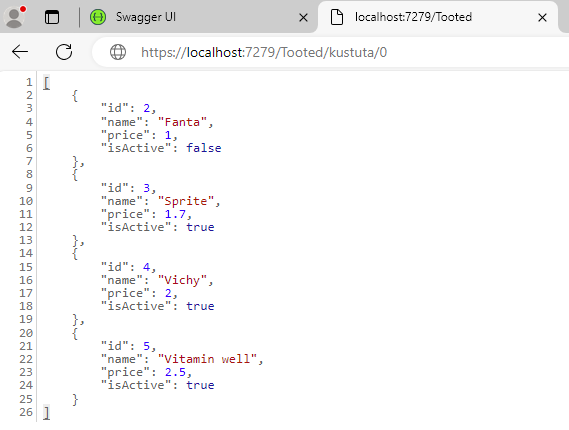
public List<Toode> Delete(int index)

{

\_tooted.RemoveAt(index);

return \_tooted;

}



Joonis . Kustutame toode index -i järgi

[HttpGet("kustuta2/{index}")]

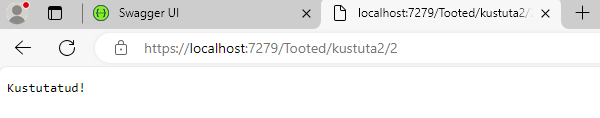
public string Delete2(int index)

{

\_tooted.RemoveAt(index);

return "Kustutatud!";

}



Joonis . Kustutame toode ja saame tulemusena Kustutatud!

[HttpGet("lisa/{id}/{nimi}/{hind}/{aktiivne}")]

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

{

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

\_tooted.Add(toode);

return \_tooted;

}



Joonis . Loome uue toode /järgi

[HttpGet("lisa")] // GET /tooted/lisa?id=1&nimi=Koola&hind=1.5&aktiivne=true

public List<Toode> Add2([FromQuery] int id, [FromQuery] string nimi, [FromQuery] double hind, [FromQuery] bool aktiivne)

{

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

\_tooted.Add(toode);

return \_tooted;

}



Joonis . loome toode läbi päringu

[HttpGet("hind-dollaritesse/{kurss}")] // GET /tooted/hind-dollaritesse/1.5

public List<Toode> Dollaritesse(double kurss)

{

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

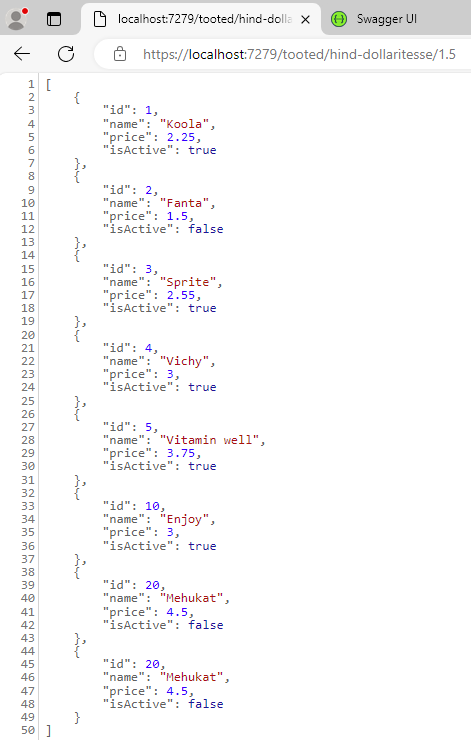
{

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

}

return \_tooted;

}



Joonis . Muudame hind korrutada määratud kurssile

// või foreachina:

[HttpGet("hind-dollaritesse2/{kurss}")] // GET /tooted/hind-dollaritesse2/1.5

public List<Toode> Dollaritesse2(double kurss)

{

foreach (var t in \_tooted)

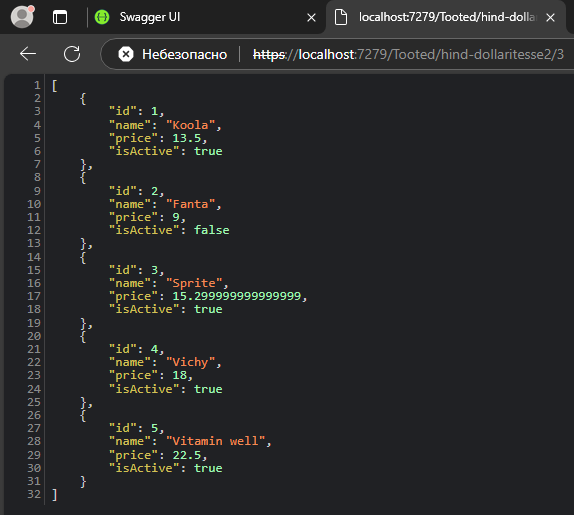
{

t.Price = t.Price \* kurss;

}

return \_tooted;

}



Joonis . Läbi foreach võib ka seda teha (kurssi meetod)

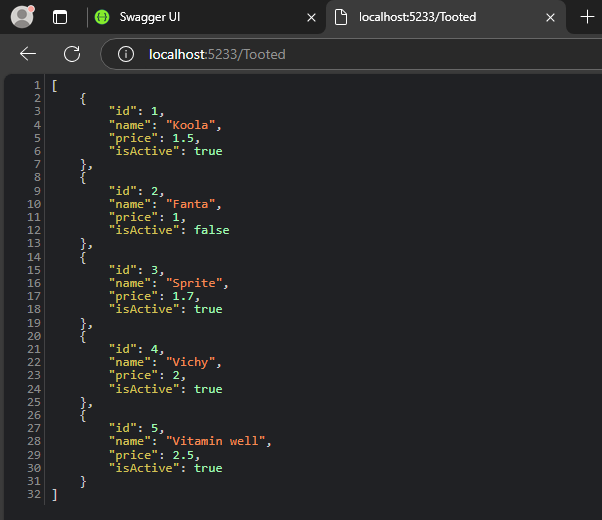
[HttpGet("kustuta-tooded")] // GET /tooted/kustuta-tooded

public List<Toode> Kustutakoik()

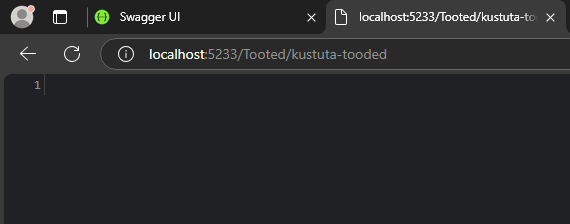
{

\_tooted.Clear();

return \_tooted;

}

Joonis . Näeme kõik tooded



Joonis . Pärast metoodi käivitamist, enam toodeid pole

[HttpGet("aktiivsus-väär")] // GET /tooted/hind-dollaritesse/1.5

public List<Toode> Aktiivsus\_vaar()

{

foreach (var t in \_tooted)

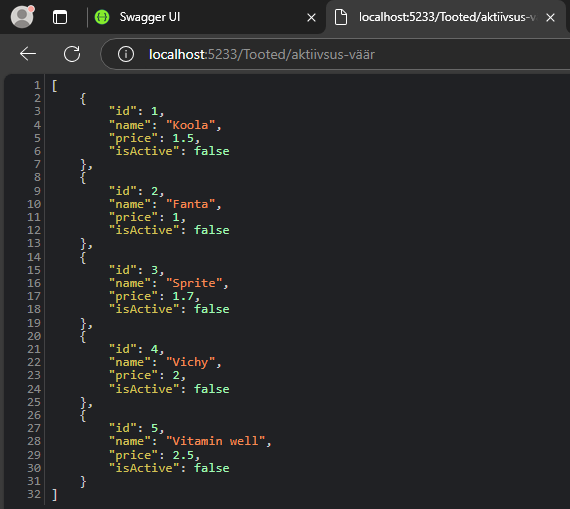
{

t.IsActive = false;

}

return \_tooted;

}



Joonis . Määrame IsActive false-le

[HttpGet("näita/{id}")] // GET /tooted/näita/id

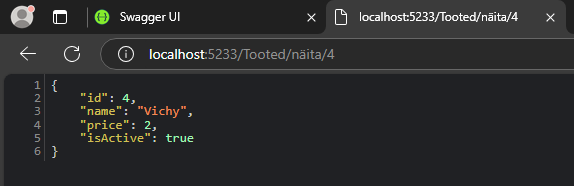
public ActionResult<Toode> Naita(int id)

{

var toode = \_tooted.Find(t => t.Id == id);

return toode != null ? toode : NotFound();

}



Joonis . Näitame toode id-ga

[HttpGet("max-hind")] // GET /tooted/max-hind

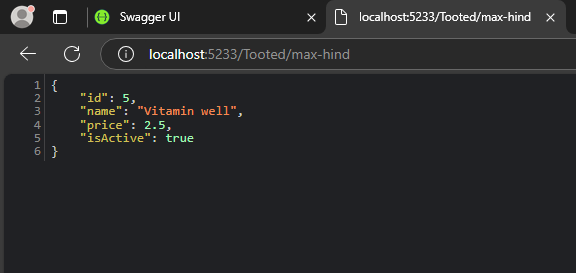
public ActionResult<Toode> Maxhind()

{

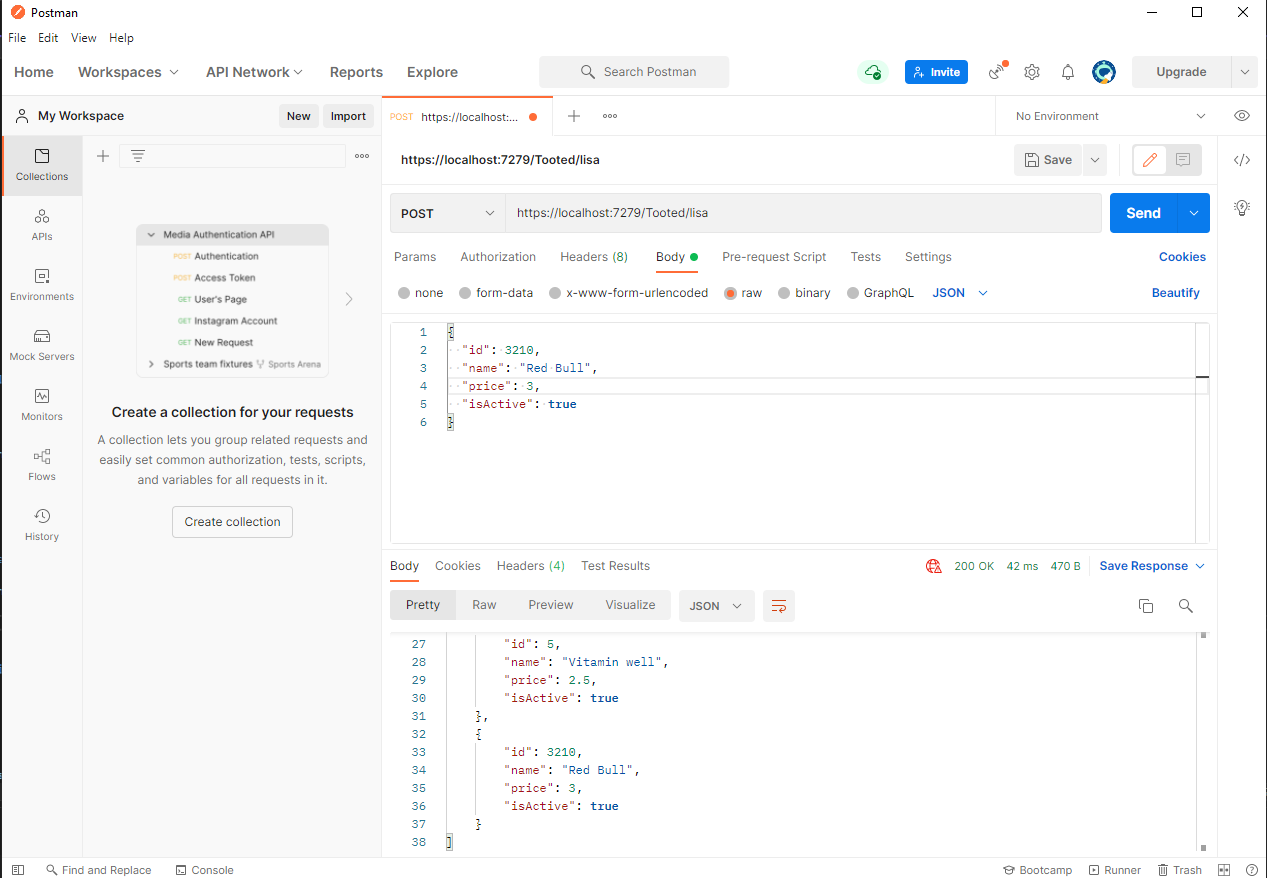
var toode = \_tooted.OrderByDescending(t => t.Price).First();

return toode != null ? toode : NotFound();

}



Joonis 22. Sorteerime kahanevalt ja näitame ainult üks andme



Joonis 23. Postmanis vajutame +, välise sisestame API leht, määrame tegevus, valime Body - raw ja tüüp JSON, sisestame andmed ja postime, kui küsib SSL -> lükkame seda tagasi ja näeme tulemuse

{

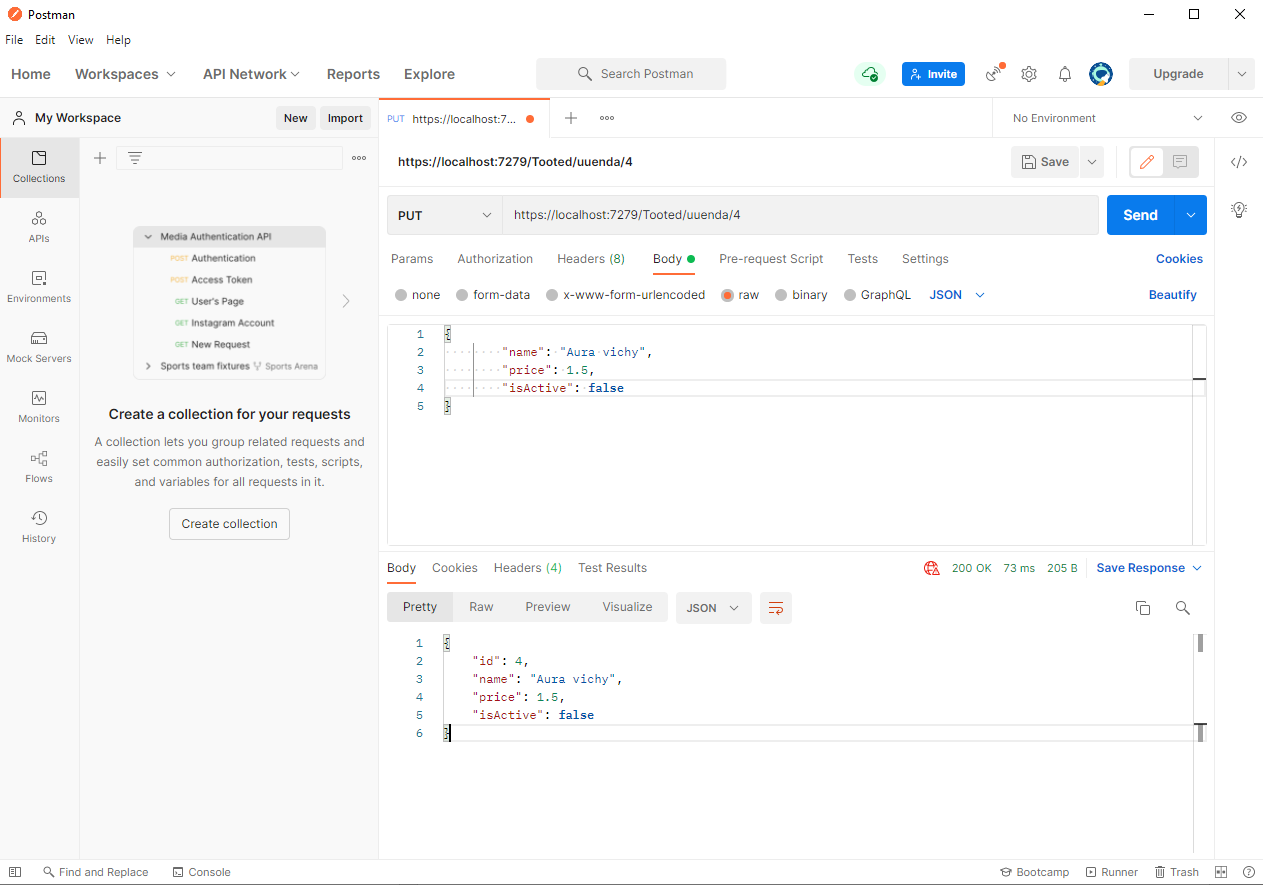
"id": 321,

"name": "Red bull",

"price": 5,

"isActive": true

}



Joonis 24. Uuendamisele , valime put, määrame lõpp lehel ID ja sisestame JSON vormis uued väärtused toodele

{

        "name": "Aura vichy",

        "price": 1.5,

        "isActive": **false**

}

<https://localhost:7279/Tooted/uuenda/4>

// PUT https://localhost:port/tooted/uuenda/{id}

[HttpPut("uuenda/{id}")]

public ActionResult<Toode> Update(int id, [FromBody] Toode updatedToode)

{

var Toode = \_tooted.Find(t => t.Id == id);

if (Toode == null)

{

return NotFound();

}

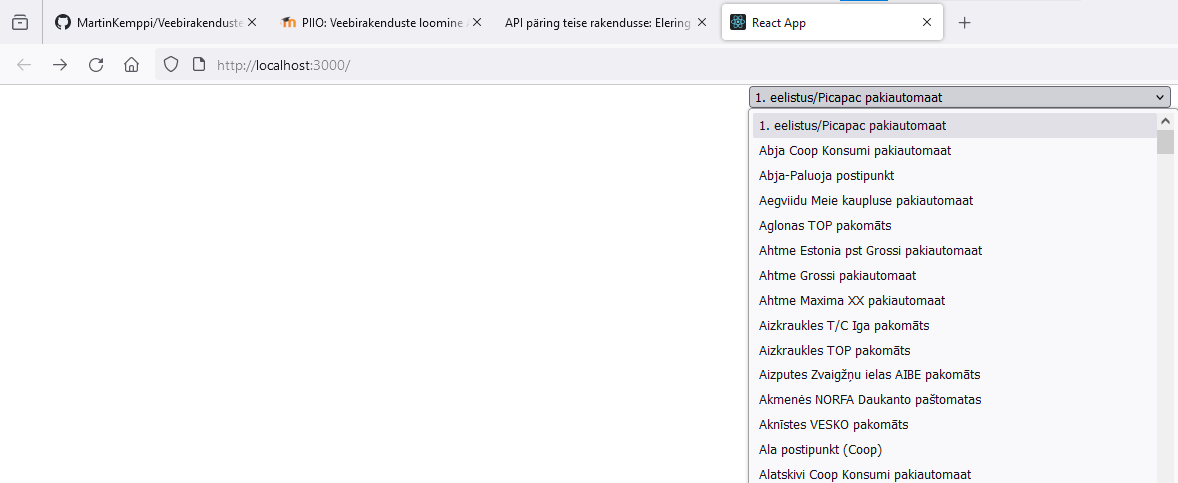
Toode.Name = updatedToode.Name;

Toode.Price = updatedToode.Price;

Toode.IsActive = updatedToode.IsActive;

return Toode;

}



Joonis 25. FrontEnd-s näeme röpploend omniva pakkiautomaadide kohad ning api.js määrame oma localhost:number

[Route("[controller]")]

[ApiController]

public class ParcelMachineController : ControllerBase

{

private readonly HttpClient \_httpClient;

public ParcelMachineController(HttpClient httpClient)

{

\_httpClient = httpClient;

}

[HttpGet("omniva")]

public async Task<IActionResult> GetParcelMachinesOmniva()

{

var response = await \_httpClient.GetAsync("https://www.omniva.ee/locations.json");

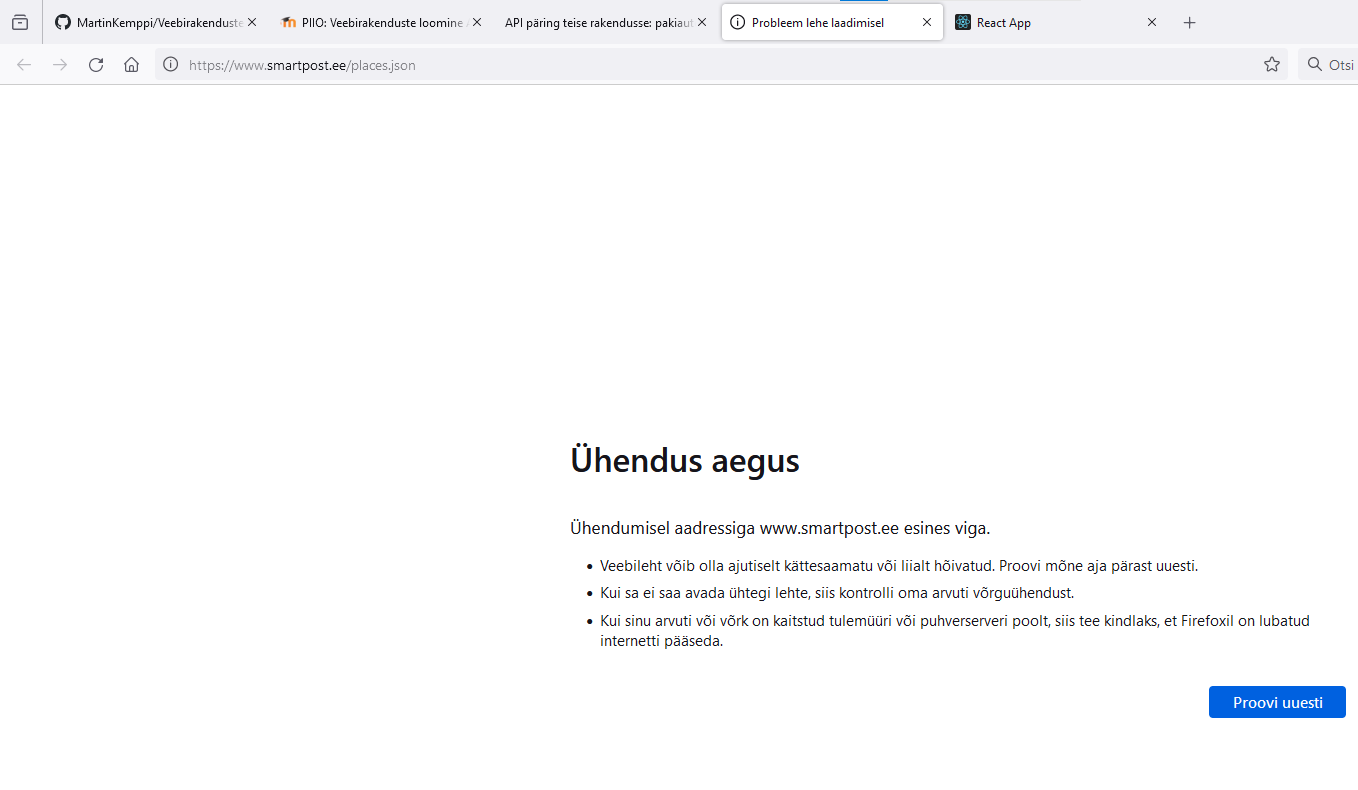
var responseBody = await response.Content.ReadAsStringAsync();

return Content(responseBody, "application/json");

}

}

https://localhost:7279/parcelmachine/omniva



Joonis 26. SmartPost ei ava ning me ei saa kuvada andmed

[Route("[controller]")]

[ApiController]

public class ParcelMachineController : ControllerBase

{

private readonly HttpClient \_httpClient;

public ParcelMachineController(HttpClient httpClient)

{

\_httpClient = httpClient;

}

[HttpGet("omniva")]

public async Task<IActionResult> GetParcelMachinesOmniva()

{

var response = await \_httpClient.GetAsync("https://www.omniva.ee/locations.json");

var responseBody = await response.Content.ReadAsStringAsync();

return Content(responseBody, "application/json");

}

[HttpGet("smartpost")]

public async Task<IActionResult> GetParcelMachinesSmartPost()

{

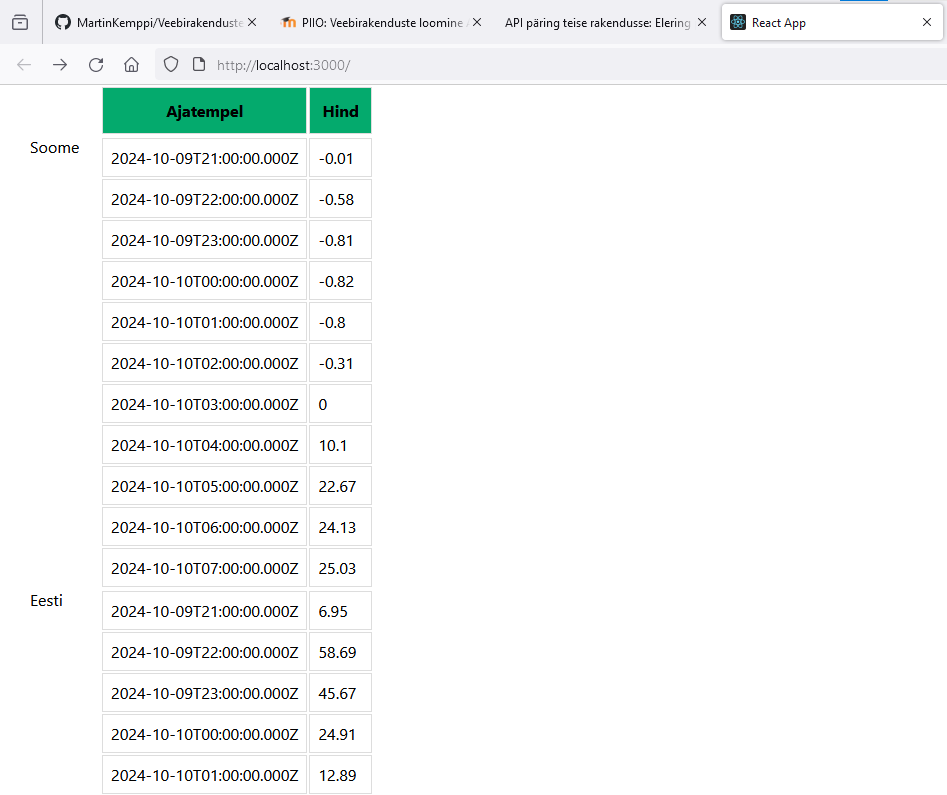
var response = await \_httpClient.GetAsync("https://www.smartpost.ee/places.json");

var responseBody = await response.Content.ReadAsStringAsync();

return Content(responseBody, "application/json");

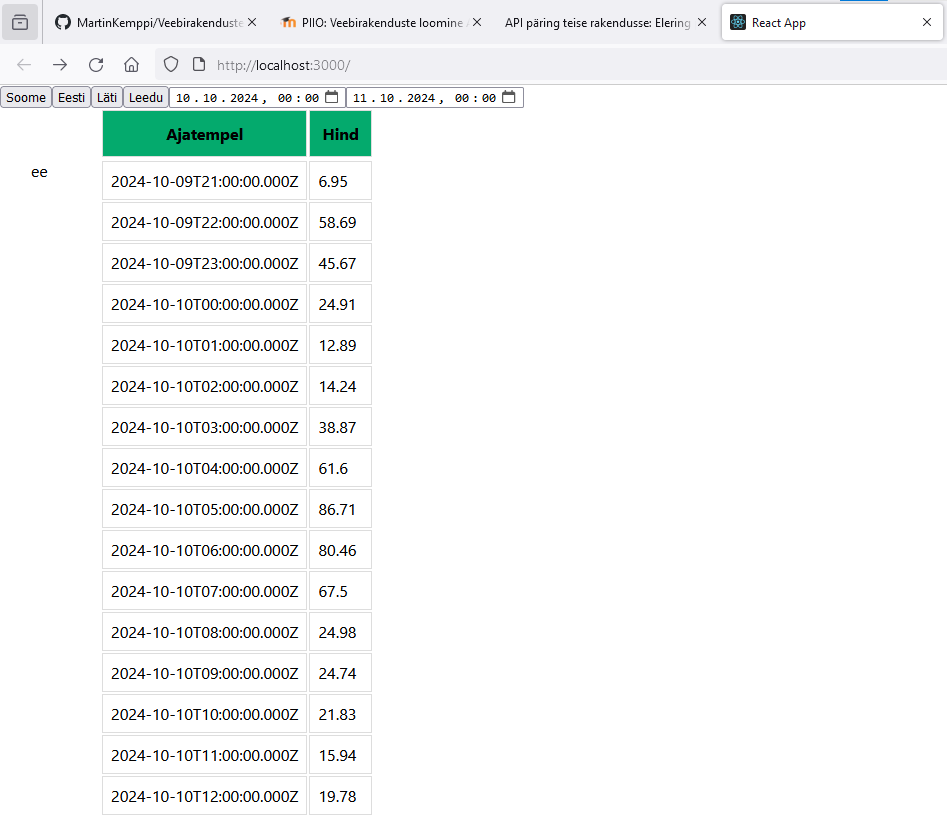
}

}



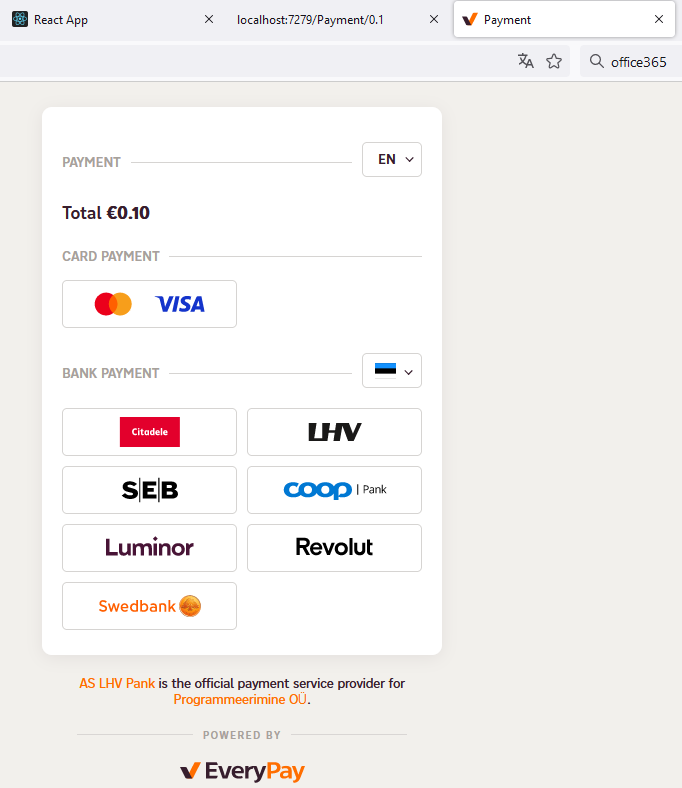
Joonis 27. Nordpool andmed

https://localhost:7279/nordpool

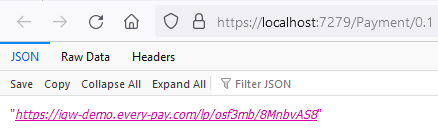


Joonis 28. Nordpool andmed valimisega: riik + päevast - päevani

https://localhost:7279/nordpool/"



Joonis 29. Käivitame päringu https://localhost:7279/Payment/0.1 ning saame lingi maksmisele, suuname sellele ja näeme tulemuse maksmisele



Joonis 30. Anname metoodile /summa maksimele ja saame pärast lingi maksmisele

private static List<Kasutaja> \_kasutaja = new()

{

new Kasutaja(1,"martinkem", "meknitram", "Martin", "Kemppi"),

new Kasutaja(2,"maksimtse", "estmiskam", "Maksim", "Tsepeleivits"),

new Kasutaja(3,"darjamil", "limajrad", "Darja", "Miljukova"),

new Kasutaja(4,"lucagluh", "hulgacul", "Luca", "Gluhhov"),

new Kasutaja(5,"oleksboha", "ohabskelo", "Oleksandr", "Bohatyrov")

};

// GET https://localhost:4444/Kasutajad

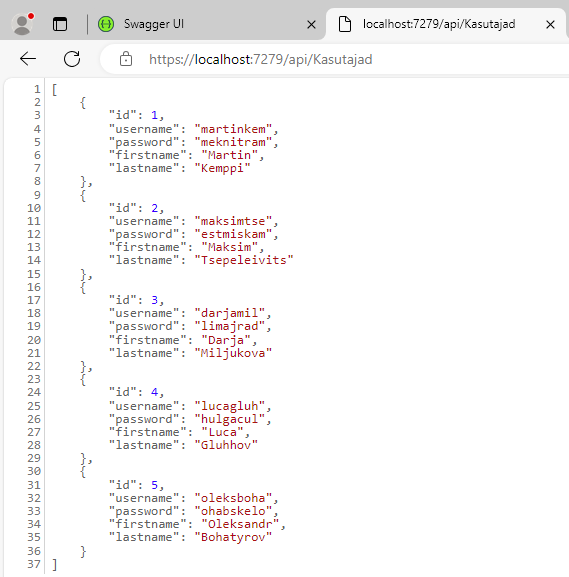
[HttpGet]

public List<Kasutaja> Get()

{

return \_kasutaja;

}

**

Joonis . Saame kõik kasutajad

// DELETE https://localhost:4444/api/kasutajad/kustuta/0

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

public List<Kasutaja> Delete(int index)

{

\_kasutaja.RemoveAt(index);

return \_kasutaja;

}

**

Joonis . Oleme kustutatud neljas index ning näeme, et 4+1 Id-ga andmet pole

// POST https://localhost:4444/api/kasutajad/lisa/1/kasutaja/parool1/mina/tema

[HttpPost("lisa/{id}/{kasutajanimi}/{salasona}/{eesnimi}/{perekonnanimi}")]

public List<Kasutaja> Add(int id, string kasutajanimi, string salasona, string eesnimi, string perekonnanimi)

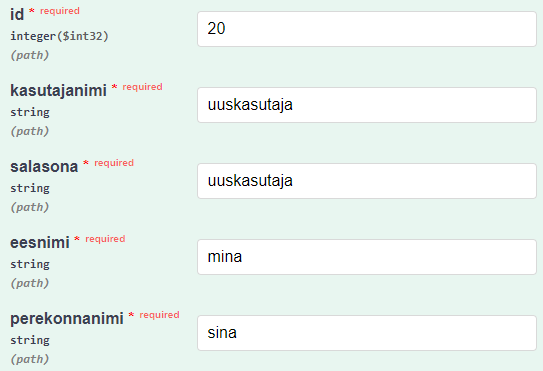
{

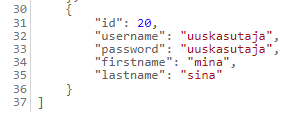
Kasutaja kasutaja = new Kasutaja(id, kasutajanimi, salasona, eesnimi, perekonnanimi);

\_kasutaja.Add(kasutaja);

return \_kasutaja;

}

**

**

Joonis . Ülevalt võetud andmed sisestamiseks oleme loonud uue kasutaja

[HttpPost("lisa2")]

public List<Kasutaja> Add2(int id, string kasutajanimi, string salasona, string eesnimi, string perekonnanimi)

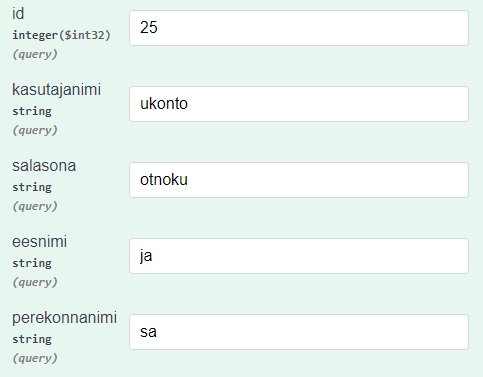
{

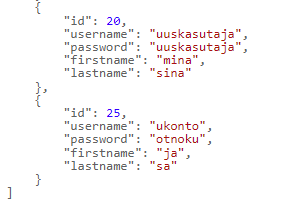
Kasutaja kasutaja = new Kasutaja(id, kasutajanimi, salasona, eesnimi, perekonnanimi);

\_kasutaja.Add(kasutaja);

return \_kasutaja;

}

**

**

Joonis . Uue kasutaja loomine

[HttpPost("lisa")]

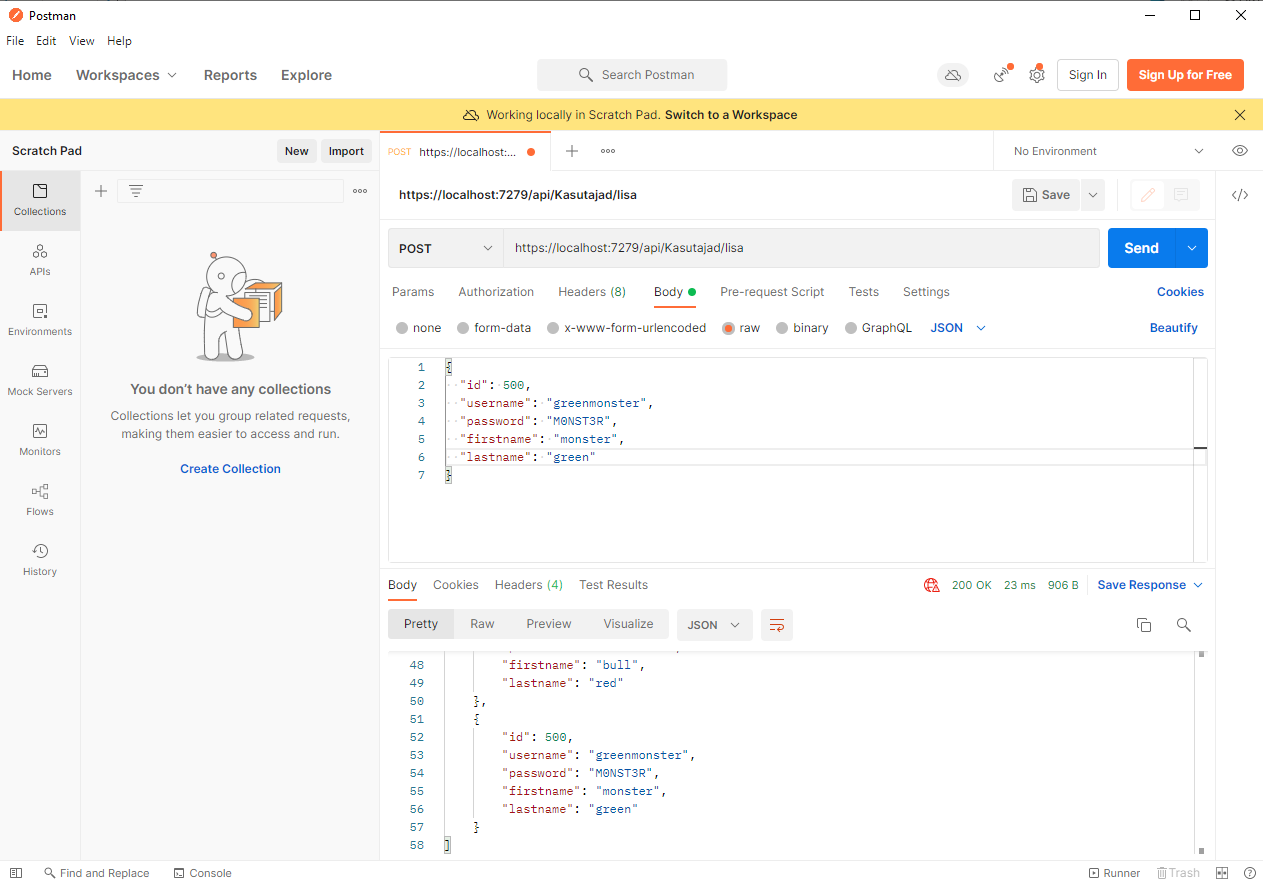
public List<Kasutaja> Add([FromBody] Kasutaja kasutaja)

{

\_kasutaja.Add(kasutaja);

return \_kasutaja;

}

**

Joonis . Lisame uue kasutaja Postmanis läbi POST JSON vormis

[HttpGet("kustuta-kasutajad")] // GET /kasutajad/kustuta-tooded

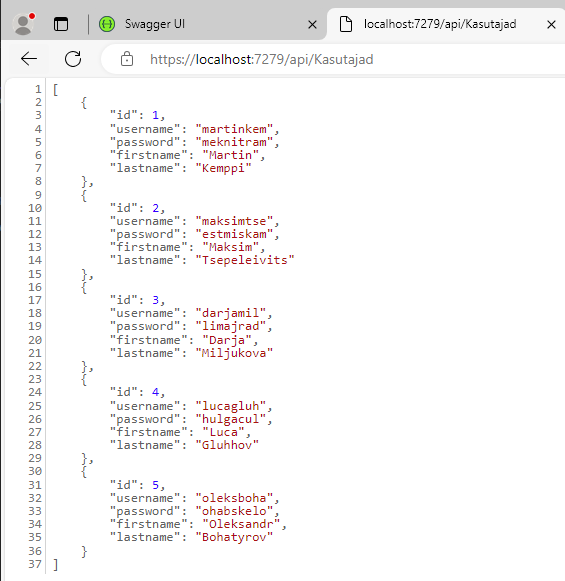
public List<Kasutaja> Kustutakoik()

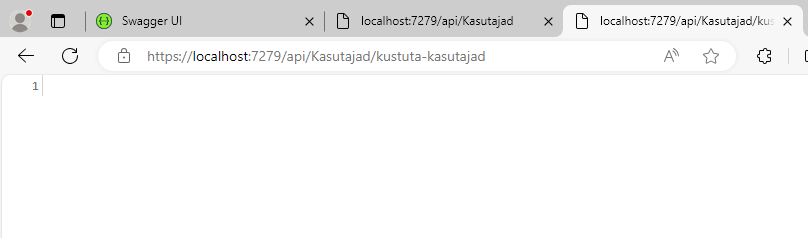
{

\_kasutaja.Clear();

return \_kasutaja;

}

**

**

Joonis . Kustutamise päringu pärast näeme, et andmeid enam pole, aga üleval on näha, et olid.

[HttpGet("näita/{id}")] // GET /kasutajad/näita/id

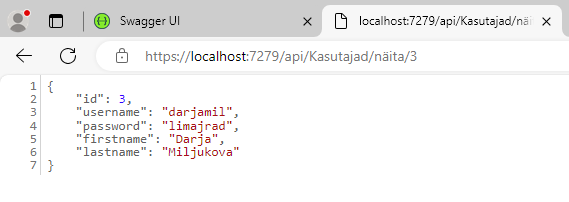
public ActionResult<Kasutaja> Naita(int id)

{

var kasutaja = \_kasutaja.Find(t => t.Id == id);

return kasutaja != null ? kasutaja : NotFound();

}

**

Joonis . Kuvame kasutaja ID-ga

// PUT https://localhost:port/kasutajad/uuenda/{id}

[HttpPut("uuenda/{id}")]

public ActionResult<Kasutaja> Update(int id, [FromBody] Kasutaja uuendatudkasutaja)

{

var Kasutaja = \_kasutaja.Find(t => t.Id == id);

if (Kasutaja == null)

{

return NotFound();

}

Kasutaja.Username = uuendatudkasutaja.Username;

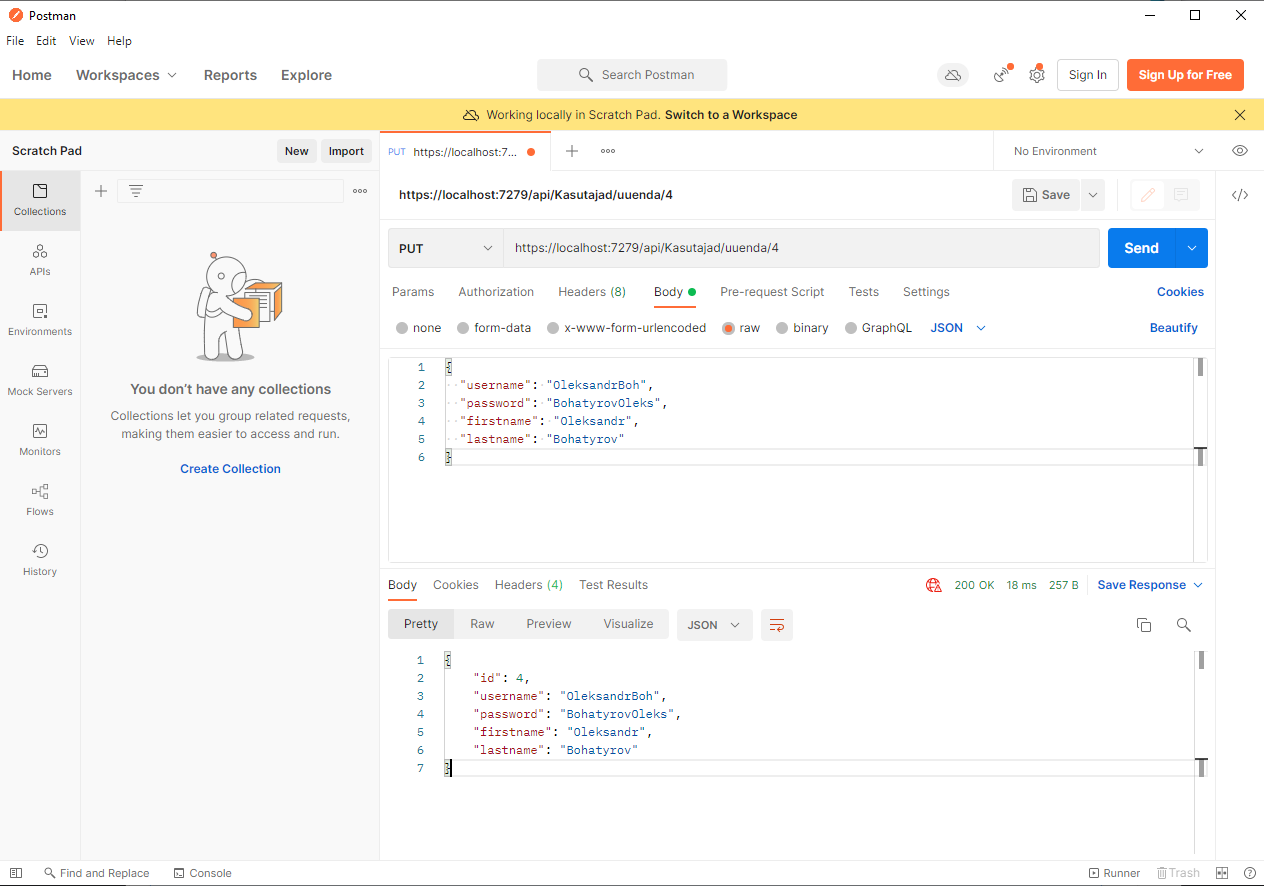
Kasutaja.Password = uuendatudkasutaja.Password;

Kasutaja.Firstname = uuendatudkasutaja.Firstname;

Kasutaja.Lastname = uuendatudkasutaja.Lastname;

return Kasutaja;

}

**

Joonis . Muudame kasutaja tema ID-st PUT käsuga Postmanis