TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Hanna Kristin Ojaveer 193693 IADB

Töögraafiku liidestuse loomine ja dokumentatsiooni kaasajastamine personalitarkvara Persona näitel

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Hanna Kristin Ojaveer

22.04.2022

Annotatsioon

Lõputöö eesmärgiks on luua töögraafiku veebiliides personalitarkvara Persona jaoks, mille abil pääsevad tarkvara kasutajad ligi töögraafiku funktsionaalsustele, ilma kasutajaliidest kasutamata. Liideses peab olema võimalik lisada, muuta, kustutada ja kuvada töögraafiku kirjeid.

Lisaks on eesmärgiks võtta kasutusele uus veebiliideste dokumenteerimise platvorm, mille kasutamine oleks mugav ja lihtne nii haldajale kui kasutajale.

Valmiv liides peab tagama, et töögraafiku kirjed oleksid kasutajatele masintöödeldaval kujul kättesaadavad ja ka muudetavad ning veebiliideste dokumentatsioon peab olema kaasaegses keskkonnas, mida on lihtne hallata ning kasutajale mugav.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 53 leheküljel, 7 peatükki, 17 joonist, 4 tabelit.

Abstract

Creating Work Schedule Interface and Updating Documentation on the Example of a Persona Personnel Software

The aim of the thesis is to create work schedule endpoints for personnel software Persona, which allows software users to access work schedule functions without using an user interface. User must be able to add, edit, delete and display work schedule entries via the endpoints.

In addition, the goal is to introduce a new platform for endpoint documentation that would be convenient and easy to use for the administrator and user.

Completed endpoints must ensure that work schedule entries are accessible and editable in a machine-readable form for the users, and the documentation must be in a modern environment that is user-friendly and easy to manage.

The thesis is in Estonian and contains 53 pages of text, 7 chapters, 17 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
ASC	*Ascending*, tõusev
CRUD	Lühend sõnadest *Create, Read, Update, Delete*
DELETE	Päring, mille tulemusel kustutatakse info andmebaasist
DESC	*Descending*, langev
DTO	*Data transfer object*, andmeedastusobjekt
GET	Päring, mis tagastab kasutajale infot andmebaasist
HTML	*Hypertext Markup Language*, veebilehtede loomiseks kasutatav keel
HTTP	*Hypertext Transfer Protocol*, veebis andmete edastamiseks loodud protokoll
JSON	*Javascript Object Notation*, standartne vorming struktureeritud andmete edastamiseks
LINQ	*Language Integrated Query*, päringukeel
POST	Päring, mille tulemusel sisestatakse info andmebaasi
PUT	Päring, mille tulemusel uuendatakse infot andmebaasis

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Igas firmas on töötajad ja mida suurem on ettevõte, seda rohkem on inimesi, kelle töötamise andmeid tuleb hallata. Persona tarkvara on loodud personalitöö arvestuseks ning raamatupidajate ja personalitöötajate tööülesannete lihtsustamiseks.

Selleks, et tarkvara muutuks paremaks ja areneks õiges suunas, saavad kliendid ehk Persona kasutajad avaldada soovi programmi funktsionaalsuste arendamiseks. Klient tõi välja, et Personas puudub hetkel liidestus, mis laseks kasutada töögraafiku funktsionaalsusi ilma kasutajaliidest kasutamata.

Lõputöö eesmärgiks ongi luua vastava personalitarkvara töögraafiku CRUD API, mis loob võimaluse edaspidi ka teiste liidestuste lisamist. Arenduse käigus uuendatakse otspunktide liidestuste dokumentatsiooni, et see oleks kasutajale mugavam ja kooskõlas loodud arendustega.

Ilma loodava lahenduseta puudub klientidel võimalus liidestamiseks ning potentsiaalsed kliendid võivad otsustada konkureeriva toote kasuks, millel on soovitav funktsionaalsus juba olemas.

2 Probleemi püstitus

Järgnevalt kirjeldatakse Persona personalitarkvara ja selle funktsonaalsusi, millele liidest looma hakatakse ning antakse ülevaade projekti eesmärgist.

2.1 Persona tutvustus

Persona V3 on veebipõhine personalitarkvara, kus on võimalik teostada kõiki peamisi personalihalduseks vajalikke toiminguid. Persona koosneb neljast moodulist – personaliarvestus, palgaarvestus, tööajaplaneerimine ja iseteenindus. Tööajaplaneerimise moodulis on kasutajal võimalik täita töögraafikuid, sisestades sinna erinevaid kirjeid - töötunde, puhkuseid, vabu päevi ja haiguslehti [1].

Joonis 1. Persona täidetud töögraafik

Joonisel on kujutatud, milline näeb välja täidetud töögraafik.

Personal on kasutusel kokku üle 130 liidese, mis edastavad masintöödeldaval kujul erinevat infot. Neist 99 tagastab erinevaid aruandeid. Kasutajal on võimalik pärida ja saata infot, aga muutmise ja kustutamise võimalust ei ole. Hetkel olemasolevas dokumentatsioonis on välja toodud olemasolevatest liidestest vaid 9. Kasutusel oleva dokumentatsiooni haldamine on ebamugav ning välimus aegunud, lisaks tuleb kasutajal lehel üles-alla liikuda, et leida üles soovitud liidese informatsioon.

Joonis 2. API dokumentatsioon hetkel

2.2 Eesmärgi püstitus

Hetkel puudub Personas liides, mille abil oleks võimalik kasutada töögraafiku funktsionaalsusi kasutajaliidest kasutamata. Lisaks on hetkel olemasoleva API dokumentatsiooni informatsioon aegunud ning kasutajale ebamugav, kuna eeldab vajaliku info leidmiseks lehel üles-alla kerimist.

Vana dokumentatsioon on hetkel teksti kujul, millel puudub filtreerimise võimalus, liidestustest ülevaate saamine on kasutajale keeruline ning välimus on aegunud ega vasta tänapäevastele tehnilistele võimalustele.

Esimene eesmärk on luua rakendusliides, mis edastaks masintöödeldaval kujul kasutajatele töögraafikute kirjete infot ning annaks võimaluse kirjeid lisada, muuta ja kustutada.

Teiseks on planeeritud dokumentatsiooni korrastamine ja liideste dokumenteerimiseks uue platvormi kasutusele võtmine. See peaks olema kaasaegne ning mugav kasutada nii haldajale kui ka tavakasutajale.

3 Liidestuse funktsionaalsus

Liidestuse põhifunktsioon on kuvada masintöödeldaval kujul töögraafiku kandeid ja nendega seonduvat informatsiooni. Andmeid loetakse andmebaasist ja teisendatakse inimloetavale kujule.

Kliendi sooviks oli laiendada rakenduse funktsionaalsust - nõuded põhinevad kliendi soovidel.

3.1 Funktsionaalsed nõuded

Kasutajapõhised funktsionaalsed nõuded, mis on teostatavad kliendi enda rakendusest:

- Kasutajana soovin saada töögraafikute kirjete informatsiooni masintöödeldaval kujul.
- Kasutajana soovin lisada töögraafiku kirjeid.
- Kasutajana soovin muuta töögraafiku kirjeid.
- Kasutajana soovin kustutada töögraafiku kirjeid.
- Kasutajana soovin näha dokumentatsioonis olemasoleva liidestuse nimekirja.
- Kasutajana soovin valida dokumentatsiooni menüüs soovitud liidese, mis suunaks lehel õigesse kohta.
- Kasutajana soovin dokumentatsioonis näha päringute näiteid, mis oleks lihtsalt loetavad.
- Kasutajana soovin dokumentatsioonis näha võimalikke parameetreid, mida on võimalik päringule kaasa anda.

Haldajapõhised funktsionaalsed nõuded:

- Dokumentatsiooni haldajana soovin, et uue liidese informatsiooni lisamine oleks loogiline ja mugav.
- Dokumentatsiooni haldajana soovin, et liidese kohta saaks lisada näiteid

3.2 Mittefunktsionaalsed nõuded

Kasutajapõhised mittefunktsionaalsed nõuded:

- Kasutaja ei pea oma tööülesannete täitmiseks kasutama veebirakendust.

Tarkvarapõhised mittefunktsionaalsed nõuded:

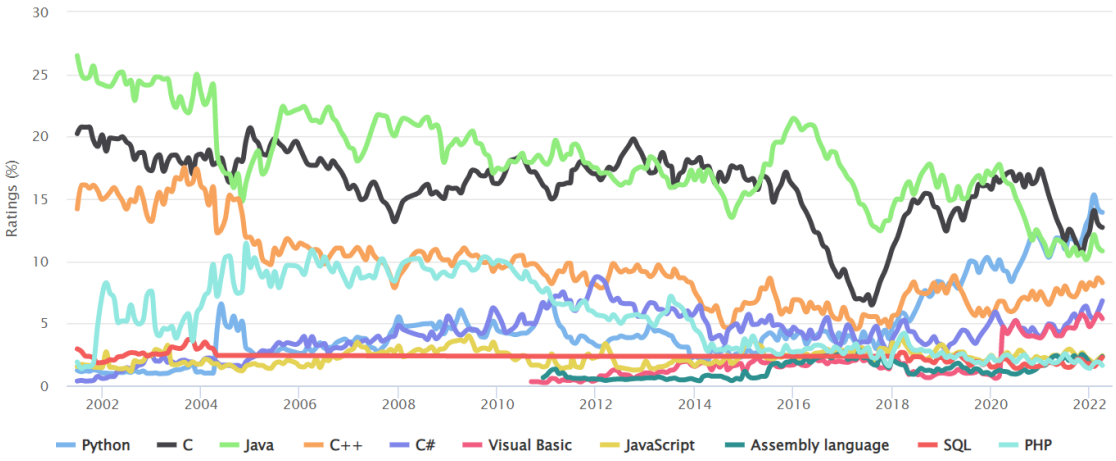
- Kuna tarkvara on kirjutatud C# keeles, siis peaks lisafunksionaaluse keel olema sama.
- Kuna tarkvara on ehitatud .NET raamistikule, siis peaks loodav lisafunktsionaalsus kasutama eelnimetatud raamistikku.
- Kuna tarkvara kasutab SQL server 2017 andmebaasi, siis peaks loodav liidestus pärima andmeid vastavast andmebaasist.
- Tarkvara testimisel kasutatakse Postman tööriista – võimalusel võiks dokumentatsiooni haldamiseks kasutada antud tarkvara võimalusi.

4 Tehnoloogiate analüüs

Veebirakenduse arendamisel on tohutult erinevaid tehnoloogiaid, mille vahel valida. Antud lõputöö raames vaadatakse üksnes suure kasutajaskonnaga tehnoloogiaid.

4.1 Programmeerimiskeel ja raamistik

Maailmas eksisteerib sadu erinevaid programmeerimiskeeli. Kuna pidevalt töötatakse välja uusi keeli, siis täpset arvu on keeruline määrata. Usutakse, et keskmiselt on neid maailmas umbes 700. Neist laialdasemalt on kasutusel umbes 50 keelt [2].



Joonis 3. Populaarseimad programmeerimiskeeled TRIOBE indeksi järgi [3]

Populaarseimate keelte hulka kuuluvad:

- Python (versioon 3.10.4) – interaktiivne objekt-orienteeritud keel, mis toetab mitut programmeerimise stiili, näiteks protseduuriline ja funktsionaalne programmerimine. Python töötab Linuxis, macOS-il ja Windowsil ja on TRIOBE indeksi järgi on hetkel kõige populaarsem keel [4].
- Java – objekt-orienteeritud programmeerimiskeel, mida interpreteerib Java Virtual Machine. See on hästi dokumenteeritud ning on maailmas laialdaselt kasutusel [5].
- C# – objekt-orienteeritud keel, millega on võimalik luua erineva keerukusega rakendusi ja seda kasutatakse .NET raamistikus. Sarnaselt eelnevalt välja toodud keeltele on C# hästi dokumenteeritud ja laialdaselt kasutusel [6].

- Javascript – objekt-orienteeritud keel, mis on rohkem tuntud veebilehtede skriptimise keelena [7].
- PHP – skriptimiskeel, mis töötati välja spetsiaalselt veebilehtede arendamiseks. Keele kood käivitatakse serveris, genereerides HTML-i, mis saadetakse sealt edasi kliendile [8].

Autor koostas tabeli, et võrrelda potentsiaalseid programmeerimiskeeli kasutades võrdluses enda kogemust ning keele õppimiskeerukust. Tabelis välja toodud keeltel on laialdaselt kasutajaid ning hästi dokumenteeritud.

Tabel 1. Programmeerimiskeelte võrdlus

Keel	**Kogemus**	**Keerukus**
Python	Hea	Madal [9]
Java	Hea	Keskmine [9]
C#	Väga hea	Keskmine
Javascript	Vähene	Keskmine [9]
PHP	Vähene	Madal [9]

Peale programmeerimiskeelte võrdlust, peab autor oluliseks koostada sarnane tabel nende keeltega seotud raamistike kohta, milles rakendust arendada. Raamistiku kasutamine võimaldab lihtsustada protsesse ja pakub tuge arhitektuuri poolelt.

Tuntumad raamistikud on [10]:

- Django – raamistik, mida kasutatakse Python keele jaoks. Raamistiku põhjal loodud rakenduste puhul on täheldatud, et rakendused on kiired, skaleeritavad, mitmekülgsed ja turvalised.
- Spring – raamistik on loodud Java platvormi jaoks ning selle abil on võimalik luua kiireid ja paindlikke süsteeme. Enamasti kasutatakse seda suure jõudlusega rakenduste loomiseks.
- .NET – raamistik on tuntud oma kiiruse ja võimsuse poolest ning on abiks dünaamiliste veebirakenduste loomisel.
- Express.js – minimaalne ja paindlik raamistik, mis on mõeldud Javascripti jaoks. Paraku suurte rakenduste loomiseks see raamistik ei sobi oma minimaalse funktsionaalsuse tõttu.

- Laravel – raamistik, mida kasutatakse PHP jaoks ning ei eelda eraldi PHP installimist, veebiserveri või muu serveri tarkvara olemasolu arendaja arvutis.

Tabel 2. Teenusepoolsete raamistike võrdlus

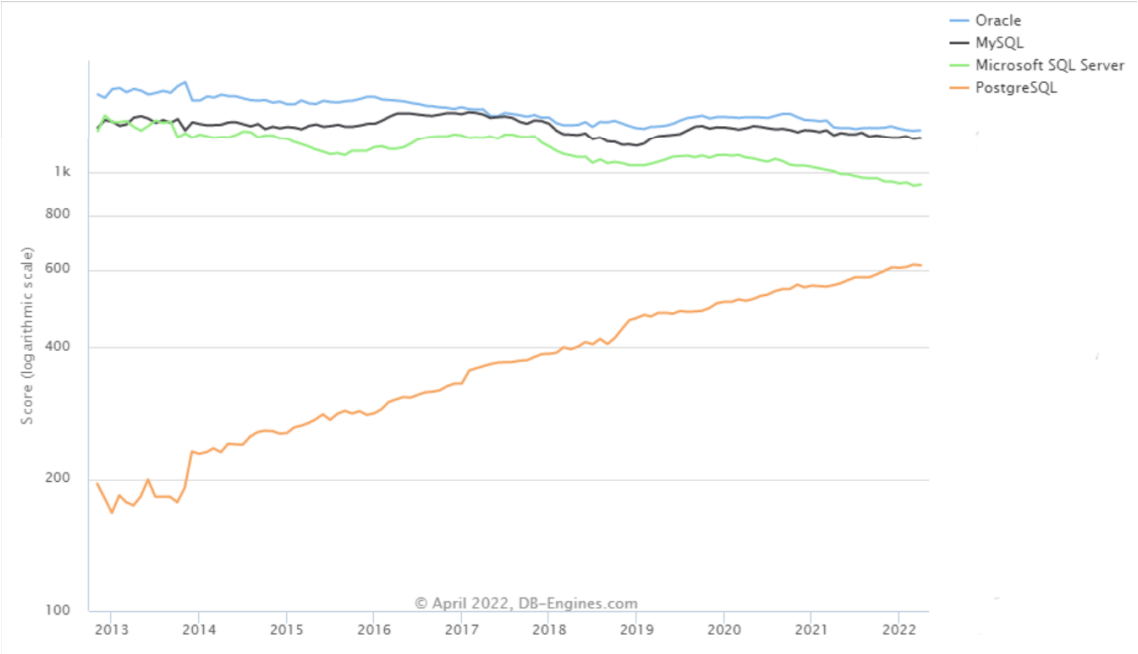
Raamistik	**Keel**	**Kogemus**
Django	Python	Puudub
Spring	Java	Keskmine
.NET	C#	Hea
Node.js	Javascript	Vähene
Laravel	PHP	Puudub

Tulemuste põhjal selgub, et mõistlik oleks arendada liidest kas Java või C# keeles, sest autoril on nendega kõige rohkem kogemust ning nii ei kulu keele õppimisele lisaaega. Samuti on autoril kogemus mõlema raamistikuga, mis on abiks rakendusliidese arendamisel.

Lõplik valik osutub C# ja .NET kasuks, sest need on laialt levinud ja hea dokumentatsiooniga. Lisaks on tarkvara, millele juurde arendusi tehakse kirjutatud eelnevalt nimetatud keeles ja raamistikus.

4.2 Andmebaas

Personalitarkvara ülesandeks on hallata ettevõtte töötajate andmeid – see tähendab, et tarkvara üheks oluliseks osaks on andmebaas, kus vastavat informatsiooni hoiustatakse. Persona puhul on tegu rakendusega, millel on palju kasutajaid, see tähendab seda, et ka hallatava info hulk on suur. Andmebaas, mida rakendus kasutab, peab olema turvaline, kiire ja vastu pidama suure hulga päringuid, mis võivad toimuda samaaegselt.



Joonis 4. Populaarsed andmebaasid DB-Engines järgi [11]

Kasutatavaimad andmebaasid on [11]:

- Oracle – andmebaas, mis toetab erinevaid andmemooduleid. Andmebaasil on tugev tehniline tugi ja põhjalik dokumentatsioon. Lisaks saab see hästi hakkama suure hulga andmete mahutamise ja töötlemisega. Paraku on Oracle tasuta versiooni funktsionaalsused väga piiratud ja tasuline versioon on üpris kallis [12].
- MySQL – relatsiooniline andmebaas, mida on kerge õppida. Individuaalselt kasutades on võimalik kasutada tasuta versiooni, mille funktsionaalsused võrreldes Oraclega on rikkalikumad. Ettevõtte jaoks kasutatav versioon on tasuline. Andmebaasi suurus on piiratud, seega andmehulga suurenedes tuleb seda silmas pidada [12].
- Microsoft SQL Server (MSSQL) – relatsiooniline andmebaas, mis tuleb hästi toime andmete salvestamise, muutmise ja haldamisega. Andmebaasil on olemas erinevad versioonid – neist viimane on SQL server 2019, millel on kõige enam funktsionaalsusi. Sarnaselt eelnevalt väljatoodud andmebaasidega, on ka sellel olemas tasuta versioon, mis on täiesti piisav väikese serveriga rakenduste loomiseks. Ettevõte jaoks kasutatav versioon on aga üks kallimaid ning kohati on litsentsi tingimused ebaselged. Lisaks on MSSQL seadistamine keeruline protsess, mis võib algajale probleeme tekitada [12].

- PostgreSQL – avatud lähtekoodiga andmebaas, mis tuleb hästi toime kasvava andmete mahuga. Sellel on suur kasutajaskond, tänu millele omab see head tuge. Andmebaasi suureks puuduseks on hetkeseisu kuvamise puudumine – pidevalt tuleb kontrollida, kas kõik töötab ikka nii nagu peab. [12]

Tabel 3. Andmebaaside võrdlus

Andmebaas	**Dokumentatsioon**	**Hind**	**Kogemus**	**Keerukus**
Oracle	Väga hea	Kõrge	Vähene	Kõrge
MySQL	Hea	Madal	Hea	Madal
Microsoft SQL Server	Väga hea	Kõrge	Hea	Kõrge
PostgreSQL	Hea	Tasuta	Hea	Madal

Alles loomisjärgus rakenduse jaoks oleks MySQL ja PostgresSQL väga head valikud, sest nende õppimine on kergem ning seadistamine lihtsam. Ettevõte, mille rakendus peab töötama suurte andmehulkadega võiks pigem valida Oracle või SQL Serveri.

Valik langeb Microsoft SQL serveri kasuks, sest autoril on sellega suurem kokkupuude. Rakendus, millele juurde arendust tehakse kasutab hetkel SQL server 2017 versiooni, millel on vähem funktsionaalsusi, kui kõige hilisemal SQL server 2019 versioonil. Edasiarenduseks oleks võimalus viia andmebaas üle uuele versioonile.

4.3 Dokumenteerimise tarkvara

API dokumenteerimise tarkvarasid, mille vahel valida, on mitmeid, vaatluse alla võetakse neist kaks - Swagger ja Postman. Neil on suur kasutajaskond ning autoril on mõlemaga juba varasem kokkupuude.

Swagger – avatud lähtekoodiga tööriist, mis automatiseerib HTTP päringute dokumentatsiooni loomist. Swaggeris on võimalik dokumentatsioonist käivitada päringuid. Lisaks on uute liidestuste loomisel vajalik teha koodile *release* ehk versiooniuuendus, et muudatused dokumentatsioonis väljenduks.

Postman – veebipõhine tööriist, millel on põhjalik dokumentatsioon koos veebiõpetustega. Tegu on intuitiivse rakendusega, mida on lihtne kasutada ja seadistada. Seda kasutatakse HTTP päringute testimiseks ning dokumenteerimiseks. Postmanil on võimalik käivitada mitmeid päringuid järjest, et kontrollida nende toimimist. [13]

Tabel 4. Dokumenteerimise tarkvarade võrdlus

Tarkvara	**Dokumentatsioon**	**Hind**	**Koodist eraldatud**	**Seadistamine**	**Keerukus**
Swagger	Hea	Tasuta	Ei	Keskmine	Keskmine
Postman	Väga hea	Tasuline	Jah	Lihtne	Lihtne

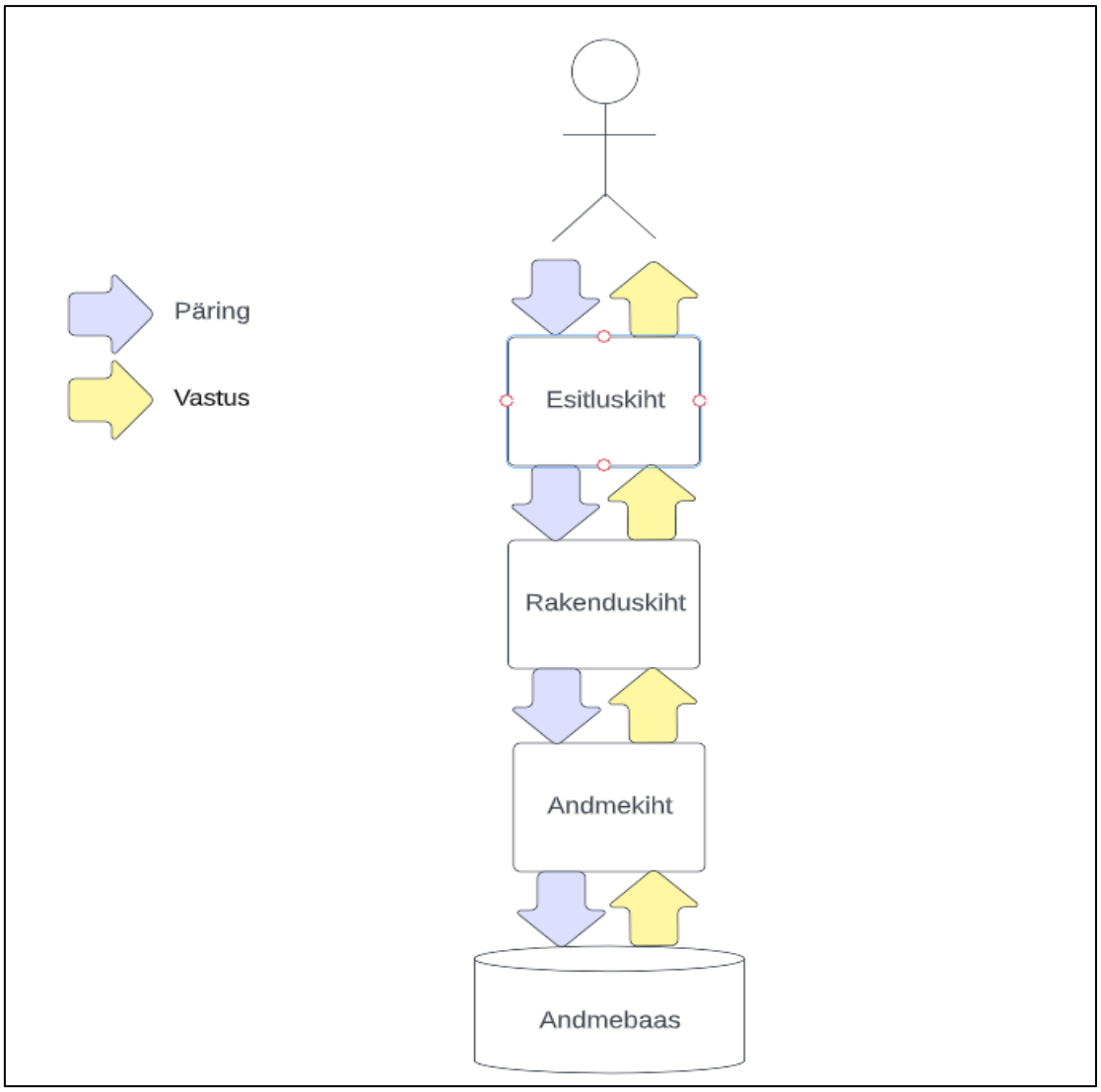
Tulemuste põhjal osutus valitavaks dokumenteerimistarkvaraks Postman, sest selle seadistamine ja kasutamine on lihtsam ning kuna tarkvara on koodist eraldatud, siis on võimalik kasutajale kuvada vaid soovitud liidestusi. Lisaks on antud tarkvara juba Personas kasutusel liideste testimisel.

4.4 Rakenduse arhitektuur

Arendatavas rakenduses on planeeritud mitmekihilise arhitektuuri kasutamine ehk esitlemise, töötluse ja andmete kättesaamine on üksteisest eraldatud. Selline mudel aitab luua korduvalt kasutatava ja paindliku tarkvara. Kui tulevikus osutub vajalikuks muudatuste tegemine, siis on neid vaja teha vaid osades kihtides, mitte kogu rakenduses.

Loodava rakenduse puhul planeeritakse kasutusele võtta kolmekihiline arhitektuur, mis koosneb andmekihist, rakenduskihist ja esitluskihist.

- Andmekiht – teostatakse päringuid andmebaasist informatsiooni saamiseks, mis tagastab saadud tulemused teistele kihtidele.
- Rakenduskiht – sellel kihil on kogu äriloogika, nagu valideerimine ja andmetega manipuleerimine.
- Esitluskiht – kiht, kust saadakse kliendilt toimingu informatsioon ja saadetakse töötlemiseks järgmisele kihile [14].



Joonis 5. Arhitektuuri mudel

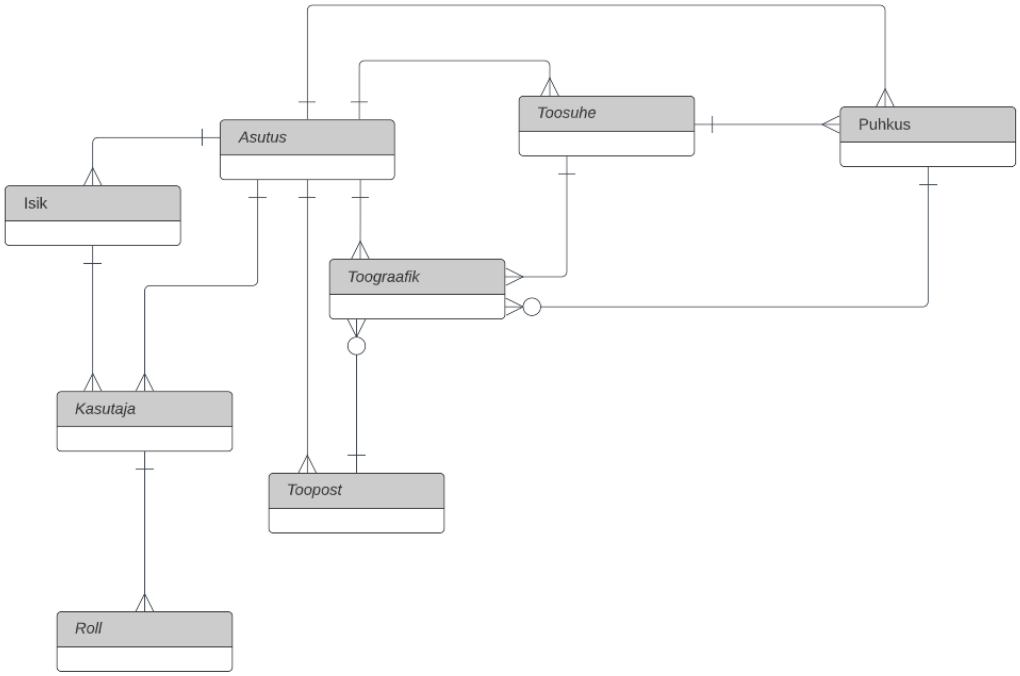
Joonis illustreerib, milline näeb välja erinevate kihtide vaheline suhtlus ja kuidas liiguvad andmed, kui klient teostab päringu.

5 Lahenduse realiseerimine

Selles peatükis antakse ülevaade tööosade realiseerimise kohta – andmebaasi mudelist, rakenduse kihtidest ning valmis lahenduse testimisest.

5.1 Andmebaasi mudel

Andmebaasi kirjeldamiseks on koostatud olemi-suhte diagramm. Kuna Persona puhul on tegu suure andmebaasiga, siis otsustas autor kuvada vaid arenduse jaoks kasutatavat andmebaasi osa. Mudelis on näidatud, kuidas on andmed omavahel seotud.



Joonis 6. Lihtsustatud andmebaasi mudel

Tabelite selgitus:

- Isik tabelis hoitakse töötaja kohta käivaid vajalikke andmeid.
- Kasutajate tabelis hoitakse tarkvara kasutajate informatsiooni ning seda, millise isikuga on kasutaja seotud.
- Rollide tabelis hoitakse kasutaja õigusi – millist informatsiooni ja funktsionaalsusi kasutaja näha või muuta tohib.
- Töösuhte tabelis hoitakse töötaja töökohaga seotud informatsiooni.
- Puhkuse tabelis hoitakse informatsiooni töötaja puhkuste kohta.

- Tööpostide tabelis hoitakse tööpostide kohta käivat informatsiooni. Kasutajal on võimalik luua Personasse kindlad tööpostid, kuhu saab märkida näiteks õhtuse vahetuse alguse ja lõppu aja ning pauside pikkuse.
- Töögraafiku tabelis hoitakse töögraafiku kirjetega seotud informatsiooni. Sellega on seotud tööposti, töösuhte ja puhkuse tabelid. Töögraafiku kandel peab olema märgitud töösuhte id, et seostada millise töötaja kohta vastav kirje käib. Puhkuse id märgitakse kirje juurde vaid siis, kui graafikus soovitakse kuvada puhkust. Tööposti id märgitakse vaid siis kui kirje käib kindla tööposti kohta.
- Kõik tabelid on seotud „Asutus“ tabeliga, sest see määrab, millise asutuse või firma juurde tabel kuulub. See on mõeldud turvalisuse tagamiseks, et kasutajal ei oleks võimalik näha mõne teise firma informatsiooni.

5.2 Andmekiht

Rakenduses loodud kihtide eesmärgiks on suhelda andmebaasiga ja teostada vajalikke toiminguid. Klassid hõlmavad andmebaasipäringuid, mis juhinduvad rakenduse funktsionaalsetest nõuetest. Päringute jaoks on võetud kasutusele päringukeel LINQ.

Andmebaasist saadud objektid muudetakse DTO-deks ehk andmeedastus objektideks, mis saadetakse edasi järgmisesse kihti. Diplomitöö koostamisel on teostatud objektide koostamine manuaalselt, et tekiks parem ülevaade, mis atribuudid ära seotakse. Arendusel kasutati kahte DTO mudelit.

```
public class ToograafikDto
{
 public long Id { get; set; }
  public DateTime Algus { get; set; }
  public DateTime Lopp { get; set; }
  public long ToosuheId { get; set; }
  public long? ToopostId { get; set; }
  public decimal? Kestus { get; set; }
  public string Markus { get; set; }
  public DateTime MuutAeg { get; set; }
  public long? PuhkusId { get; set; }
  public KlVaartusDto PuhkusLiikKl { get; set; }
 }
```

Joonis 7. Kasutajale informatsiooni kuvamiseks kasutatav DTO

```
public class ModifyToograafikDto
{
  public DateTime Algus { get; set; }
  public DateTime Lopp { get; set; }
  public long ToosuheId { get; set; }
  public long? ToopostId { get; set; }
  public decimal? Kestus { get; set; }
  public string Markus { get; set; }
  public string PuhkusLiikKl { get; set; }
}
```

Joonis 8. Kasutaja poolt saadetava informatsiooni päringu DTO

ToograafikDto on kasutusel kasutajale informatsiooni kuvamiseks ja ModifyToograafikDto kasutaja käest informatsiooni küsides.

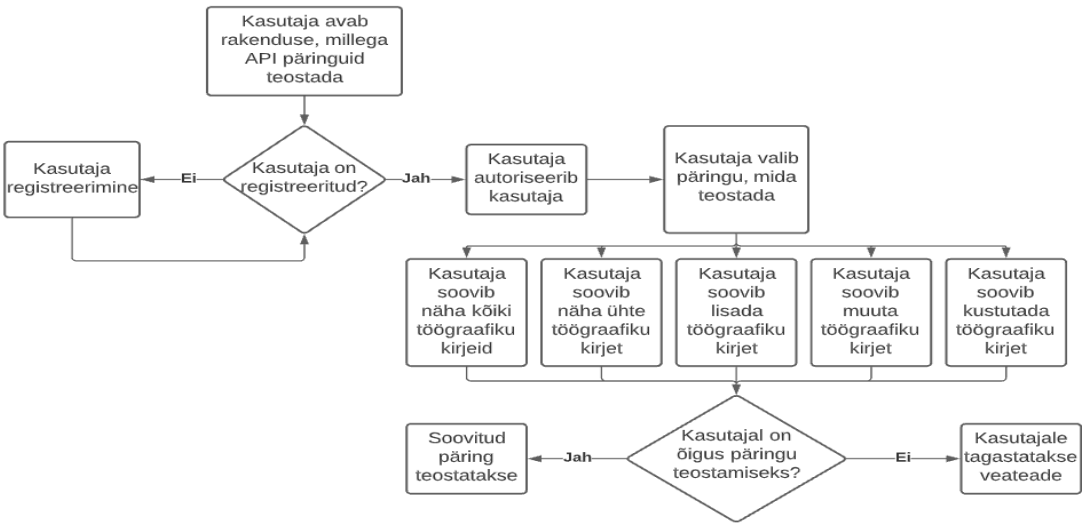
Selleks, et andmeid muuta või kätte saada on vaja saata andmebaasi vastavad päringud. Lisas 2 ja 3 on toodud välja vastavalt töögraafiku kirje info pärimine ja kirje kustutamise kohta koodi näited.

5.3 Rakenduskiht

Rakenduse äriloogika paremaks selgitamiseks on kasutatud koodi ja seda visualiseerivat plokkskeemi.

5.3.1 Üldine

Tegemist on liidese ülesehituse plokkskeemiga, mis näitab samme, mida kasutaja peab tegema, et loodavat funktsionaalsust kasutada ning millised on võimalikud valikud.

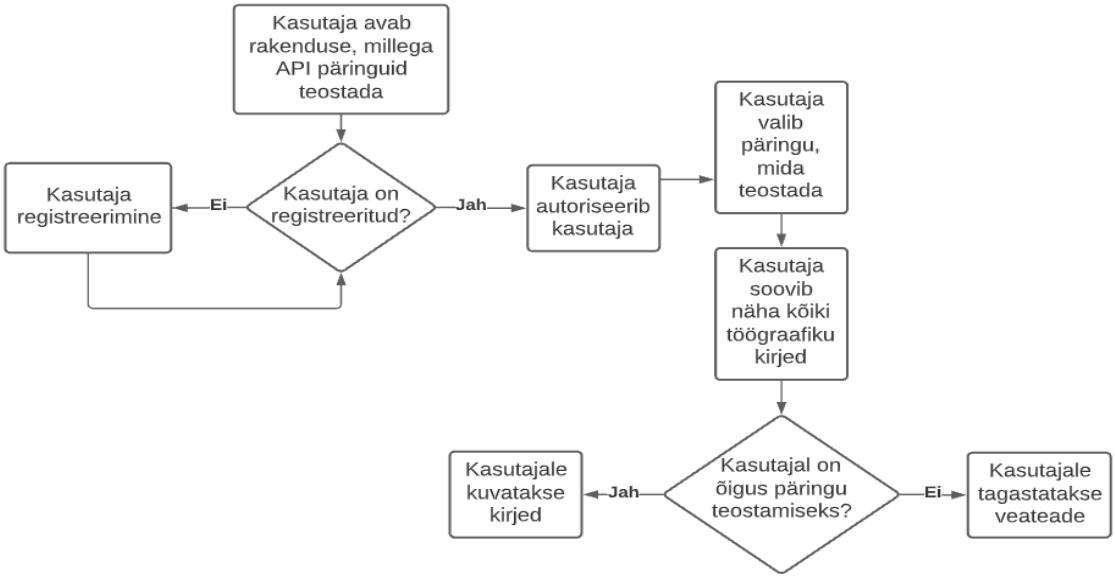


Joonis 9. Töögraafiku kirjete päringute üldine plokkskeem

Joonisele vastav kood välja toodud Lisas 4.

5.3.2 Kõigi töögraafiku kirjete kuvamine

Funktsionaalsus on vajalik selleks, et klient saaks kasutada kõigi töögraafiku kirjete informatsiooni masintöödeldaval kujul.



Joonis 10. Töögraafiku kirjete pärimise plokkskeem

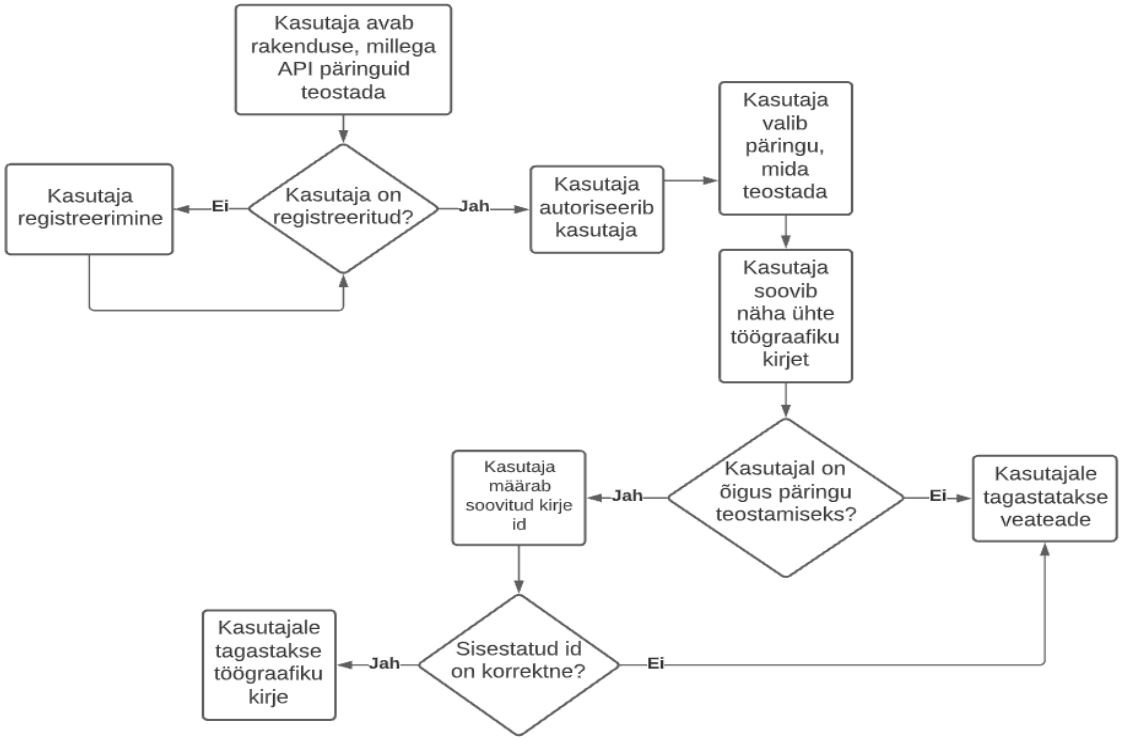
Joonis näitab, millised sammud tuleb töögraafiku kirjete kuvamiseks läbi teha. Järgneb vastav kood:

```
[HttpGet]
public GetAllToograafikudResponse GetAllToograafikud()
{
    // Õiguste ja filtrite kontroll
    ...
    try
    {
        var result = GraafikudServices.GetToograafikudForApi(alates,
kuni, eAlatesKuniSubjekt, limiit, eJarjestus, eJarjestusSubjekt,
nihe, toosuheIds);
        var response = new GetAllToograafikudResponse
        {
            Parameeterid = ParametersData(toosuheIds, alates, kuni,
alatesKuniSubjekt, limiit, jarjestus, jarjestusSubjekt, nihe,
result.TotalResult),
            Tulemus = result.Result,
            Teated = NotificationsList( nihe, result.TotalResult,
limiit)
        };
        return response;
    }
    ...
    // Veahaldus
}
```

Joonis 11. GET Töögraafiku kirjete päringu koodi näidis

5.3.3 Töögraafiku kirje kuvamine

Funktsionaalsus on vajalik selleks, et klient saaks vaadata kindla töögraafiku kirje informatsiooni, ilma et ta peaks kõiki kirjeid läbi sirvima.



Joonis 12. Töögraafiku kirje päringu plokkskeem

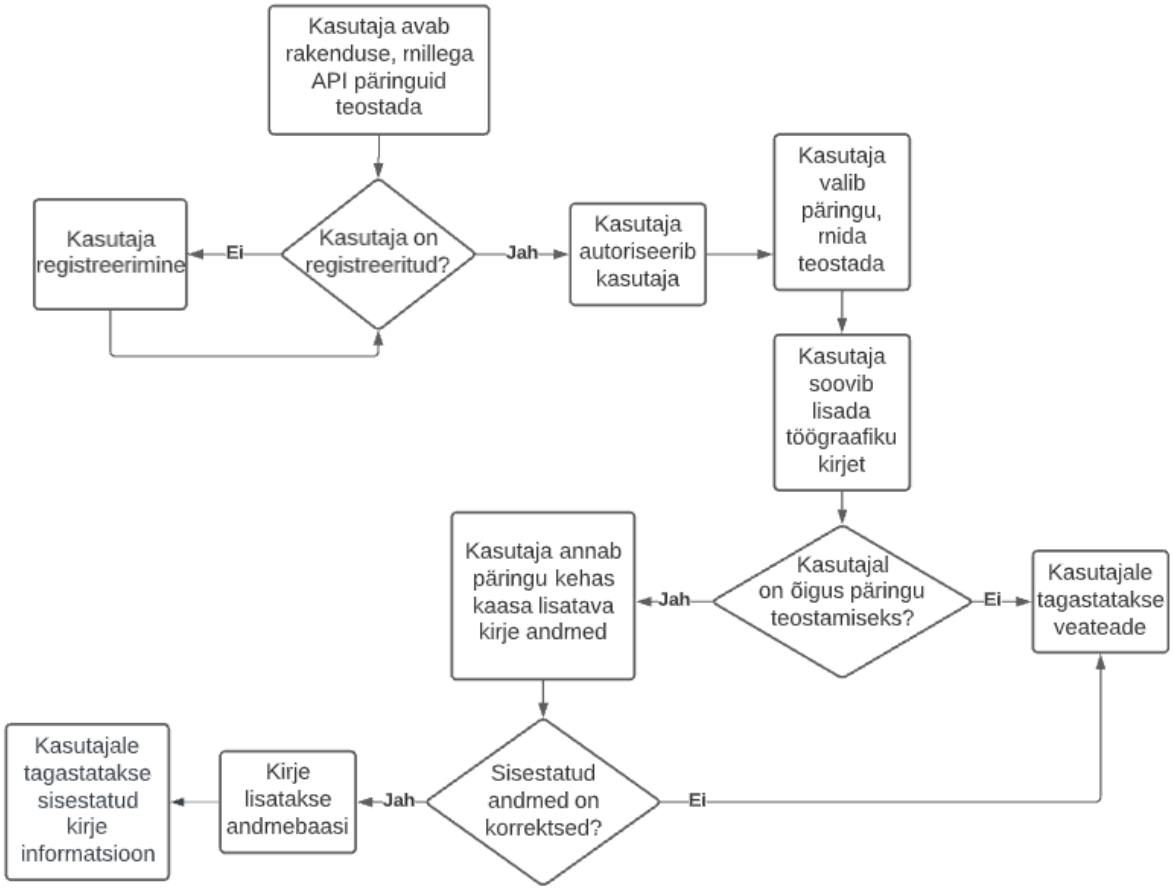
Joonis näitab, millised sammud tuleb töögraafiku kirje kuvamiseks teha. Järgneb vastav kood:

```
[HttpGet]
public ToograafikDto GetToograafik(long id)
{
    UserRoleCheck(SecurityRole.ApiToograafikudVaatamine);

    try
    {
        var toograafik = GraafikudServices.GetToograafikForApi(id);
        if (toograafik == null)
            throw
CreateHttpResponseException(ErrorMessages.SisendiViga,
                HttpStatusCode.NotFound);
        return toograafik;
    }
     ...
     // veahaldus
}
```

Joonis 13. GET Töögraafiku kirje päringu koodi näidis

5.3.4 Töögraafiku kirje lisamine

Funktsionaalsus on vajalik selleks, kasutaja saaks panna paika töötajate tööl olemise ajakava ning puhkuste graafikud. Plaani saab täita lisades graafikusse vastavaid kirjeid.

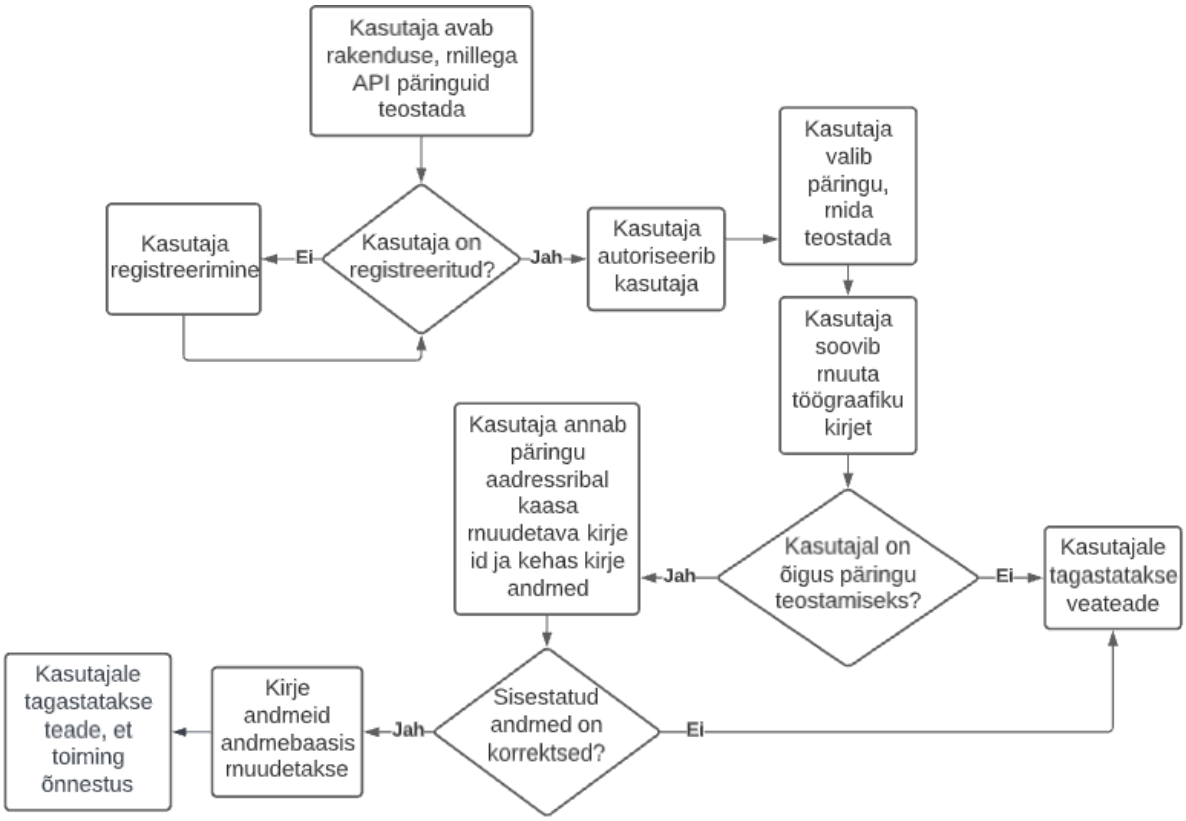


Joonis 14. Töögraafiku kirje lisamise plokkskeem

Joonis näitab, tegevusi töögraafiku kirje lisamiseks. Vastav kood on toodud Lisas 4 olevas üldkoodis ja meetodi nimetuseks on LisaToograafik.

5.3.5 Töögraafiku kirje muutmine

Funktsionaalsus on vajalik selleks, et kasutaja saaks muuta kirje andmeid juhul kui lisamisel on tehtud viga või on vaja andmeid parandada muul põhjusel.

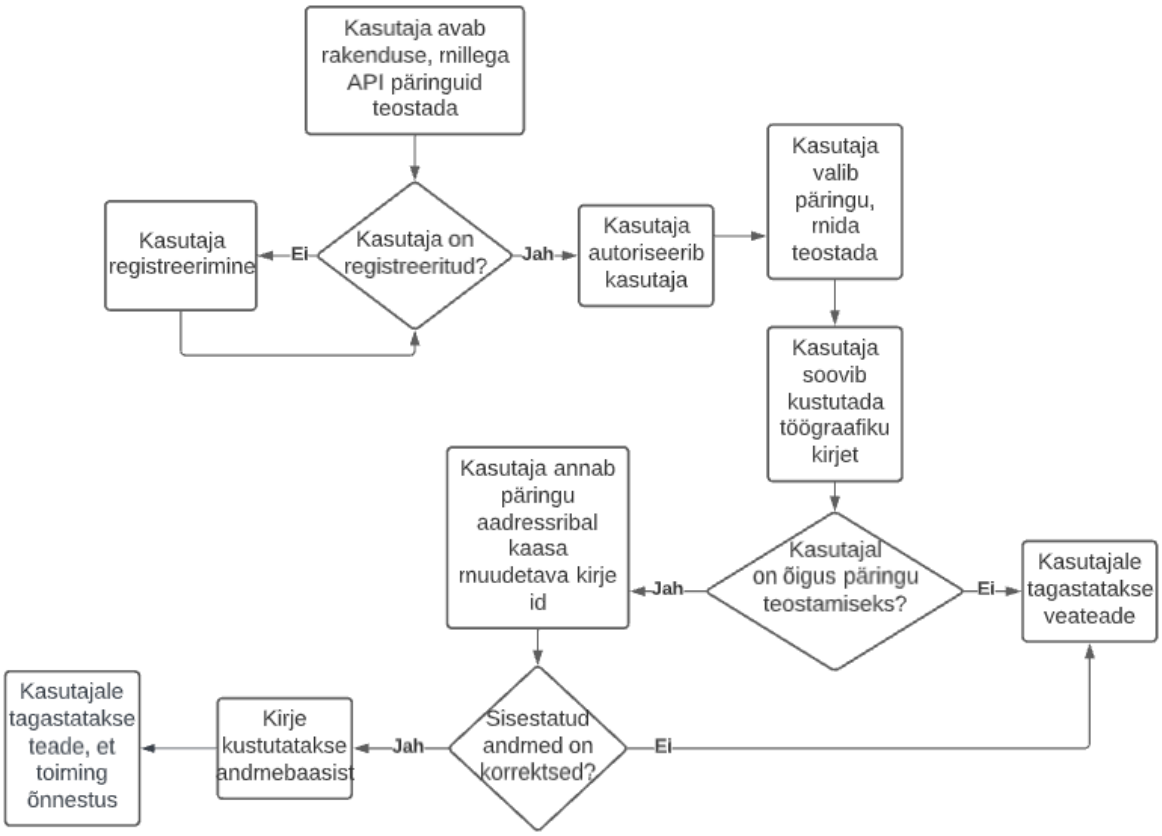


Joonis 15. Töögraafiku kirje muutmise plokkskeem

Joonis näitab, millised sammud tuleb teha töögraafiku kirje muutmiseks. Vastav kood on toodud Lisas 4 olevas üldkoodis ja meetodi nimetuseks on UpdateToograafik.

5.3.6 Töögraafiku kirje kustutamine

Funktsionaalsus on vajalik juhul, kui kasutaja on kogemata sisestanud mõne kirje valele päevale või on mõnel muul põhjusel vaja see töögraafikust eemaldada.



Joonis 16. Töögraafiku kirje kustutamise plokkskeem

Joonis näitab, mida tuleb teha töögraafiku kirje kustutamiseks läbi teha. Järgneb vastav kood:

```
[HttpDelete]
public DeleteResponse DeleteToograafik(long id)
{
    UserRoleCheck(SecurityRole.ApiToograafikudMuutmine);

    try
    {
        GraafikudServices.DeleteToograafikApi(id);
        return new DeleteResponse();
    }

    ...

    //veahaldus
}
```

Joonis 17. DELETE Töögraafiku kirje päringu koodi näidis

5.4 Esitluskiht

Esitluskihti eesmärgiks on tegeleda kliendi poolt teostatud HTTP päringutega ja vastuse tagasi saatmisega.

API kontrolleri puhul tuleb:

- Rakendatakse BaseApiController klass, mis on Personas kasutusel olev klass, kus on üldisemad liidestega teostatavad meetodid – see omakorda on seotud ApiControlleriga
- ApiController klassis on HTTP päringute töötlemise jaoks vajalikud meetodid ja atribuudid defineeritud.
- Iga defineeritud meetodi juurde tuleb lisada annotatsioon, mis määrab, mis tüüpi päringut meetod täitma peab ning milliseid parameetreid see vajab.

5.5 Testimine

Loodud liideseid testiti manuaalselt. Testimise eesmärgiks oli kinnitada, et päringud tagastavad oodatud tulemuse ning kontrollida, et kasutaja ei näeks informatsiooni, mida ta näha ei tohiks – näiteks mõne teise kliendi andmeid. Pärigu vastuste kontrollimisel tuginetakse töögraafiku veebirakenduses kuvatavale töögraafiku vaatele.

Testimisel osalevad arendajad, testijad ja funktsionaalsuse tellinud klient. Liidest on võimalik testida kasutades Postmani tööriista.

Testimisel kontrollitavad funktsionaalsused:

- Töögraafiku kirjete andmed on korrektsed
- Kasutaja saab teostada päringuid vaid siis, kui tal on päringu tegemiseks õigused
- Kasutajale kuvatakse vaid tema töögraafikute kirjeid
- Töögraafiku kirjete lisamisel, muutmisel või kustutamisel peavad muudatused olema kajastatud ka veebirakenduses kuvatavas töögraafikus.

6 Loodud lahendus

Lõputöö tulemusena loodi 4 töögraafiku liidest – kuvamine, lisamine, muutmine ja kustutamine.

6.1 Kuvamine

Kuvamisel on kaks valikut – ühe või kõigi kirjete kuvamine. Seda saab määrata, kui lisada aadressribale soovitud kirje id. Kuna vastavalt kasutajale võib töögraafiku kirjete hulk varieeruda 1000 kuni üle 10 000 vahel, siis on lisatud päringule ka filtrid, et kasutaja saaks näha just soovitud kirjeid.

Filtris on võimalik määrata:

- ToosuheId – võimalik sisestada mitu töösuhteId-d kui need on eraldatud komaga. Pole kohustuslik.
- Järjestus – võimalikud väärtused „Asc“ ja „Desc“. Töögraafiku kirjed järjestatakse kirje algus kuupäeva ja aja põhjal. Vastavalt valikule kuvatakse esimesena kas kõige vanemat töögraafiku kannet või kõige uuemat. Võib jätta tühjaks.
- Alates – võimalik sisestada kuupäev, mis ajast alates kirjeid näha soovitakse. Võib jätta tühjaks.
- Kuni – võimalik sisestada kuupäev, millise ajani kirjeid näha soovitakse. Võib jätta tühjaks.
- Limiit – piirab kuvatavate tulemuste arvu. Peab jääma alla 1000. Vaikimisi on väärtuseks määratud 1000.
- Nihe – kuna kuvatavate tulemuste arv on piiratud, siis järgmiste andmete kuvamiseks on võimalik suurendada nihet. Siis on võimalik näha varasemalt peidus olevaid andmeid. Võib jätta tühjaks.

Töögraafikute ja töögraafiku näidispäringud on toodud välja vastavalt Lisas 5 ja Lisas 6.

6.2 Lisamine

Lisamisel antakse päringusse kaasa JSON formaadis informatsioon.

Päringus on väljad:

- Algus – määrab töögraafiku kirje algusaja. Formaat YYYY-MM-DDTHH:MM:SS. Kohustuslik.
- Lõpp – määrab töögraafiku kirje lõppaja. Formaat YYYY-MM-DDTHH:MM:SS. Kohustuslik.
- TöösuheId – määrab seose töögraafiku kirje ja töösuhte vahel. Kohustuslik.
- TööpostId – kui kirje käib kindla tööposti kohta, siis saab seda määrata sisestades tööpostId. Võib jätta tühjaks.
- Kestus – määrab kirje kestuse. Näiteks 8 tundi. Võib jätta tühjaks.
- Märkus – kirje juurde on võimalik lisada märkuseid, mida kuvatakse töögraafikus. Võib jätta tühjaks.
- PuhkusliikKl – kui töögraafikusse soovitakse sisestada puhkust, siis tuleb määrata puhkuseliik. Näiteks õppe-, vanema- või põhipuhkus.

Lisamise näidispäring on toodud välja Lisas 7.

6.3 Muutmine

Muutmisel antakse aadressribal kaasa kirje id, mida soovitakse muuta, lisaks antakse päringukehas kaasa JSON formaadis informatsioon. Päringuväljad on kirjeldatud lisamise päringu all.

Muutmise näidispäring on toodud välja Lisas 8.

6.4 Kustutamine

Kustutamisel antakse aadressribal kaasa kirje id, mida soovitakse kustutada. Kustutamise õnnestumisel kuvatakse vastav teade.

Kustutamise näidispäring on toodud välja Lisas 9.

6.5 Dokumentatsioon

Dokumenteerimiseks võeti kasutusele Postman tarkvara, kuhu toodi üle kõik varasemast olemasolev dokumentatsioon ja lisati juurde loodud liideste informatsioon. Nüüd on haldajal võimalik mõne minutiga lisada dokumentatsiooni vajalikud muudatused.

Kasutajal on nüüd võimalik menüüs näha kõiki liideseid, mis on sisestatud ning soovitud liidesele vajutades suunatakse ta otse vastava informatsiooni juurde. Lehekülg on kaasaegsem ning näited arusaadavamad ja kasutajasõbralikumad.

Dokumentatsiooni näide on toodud Lisas 10.

7 Kokkuvõte

Lõputöö tulemusena valmis personalitarkvara töögraafiku CRUD API, mis loob võimaluse edaspidi ka teiste liidestuste lisamist. Rakenduse töökindlust testiti Postman tarkvara abil. Arenduse käigus uuendati ühenduspunktide liidestuste dokumentatsiooni, mis on nüüd kaasajastatud ja kooskõlas loodud arendusega.

Tänu tehtud tööle on võimalik liidestada uusi kliente Personaga ja koheselt testida, kas liidestuse funktsionaalsus vastab nõuetele. Sellega tõusis klientide rahulolu, mis kindlustab uute klientide lisandumise ja olemasolevate klientide lojaalsuse suurenemise.

Hetkel on võimalik lisada, muuta ja kustutada vaid ühe töögraafiku kirje kaupa. Kui liides on olnud mõnda aega kasutajatele kättesaadav ja saab olla kindel, et kõik töötab korrektselt, siis on võimalik teha juurde arendus, millega eelnimetatud toiminguid saaks teha mitmetele kirjetele korraga.

Dokumentatsioonis on võimalik leida lahendus, kuidas oleks mõistlik kuvada kõiki erinevaid aruannete päringuid nii, et kasutaja saaks täieliku ülevaate. Kuna aruannete päringud on loodud aastaid tagasi, siis enne avalikustamist tuleks kontrollida, kas kõik päringud toimivad vastavalt ettenähtud funktsionaalsusele. See ei mahtunud hetkel lõputöö skoopi.

Kasutatud kirjandus

[1] „Personalitarkvara Persona V3,“ https://www.rmp.ee/tarkvara/palk/personav3/personalitarkvara-persona-v3-2018-03-19 [WWW]

[2] „How Many Coding Languages Are There?“ https://breakingintostartups.com/how-many-coding-languages-are-there/ [WWW]

[3] „TIOBE Index for April 2022,“ https://www.tiobe.com/tiobe-index/ [WWW]

[4] „General Python FAQ,“ https://docs.python.org/3/faq/general.html#what-is-python [WWW]

[5] „Java: Everything a Beginner Needs to Know,“ https://www.coursereport.com/blog/what-is-java-programming-used-for [WWW]

[6] „A tour of the C# language,“ https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ [WWW]

[7] „About JavaScript,“ https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript [WWW]

[8] „What is PHP?, “ https://www.php.net/manual/en/intro-whatis.php [WWW]

[9] „List Of Top Programming Languages From Easy To Hard To Learn,“ https://codeandhack.com/easy-to-hard-to-learn-programming-languages/ [WWW]

[10] „10 Popular Web Frameworks for Web App Development in 2022,“ https://www.monocubed.com/blog/most-popular-web-frameworks [WWW]

[11] „DB-Engines Ranking - Trend Popularity,“ https://www.monocubed.com/blog/most-popular-web-frameworks [WWW]

[12] „Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others,“ https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/ [WWW]

[13] „REST API Testing Part 1: Get Started with Postman, Swagger, and Assertible,“ https://gorillalogic.com/blog/rest-api-testing-part-1-get-started-with-postman-swagger-and-assertible/ [WWW]

[14] „Andmebaasi arhitektuur,“ https://et.tutorialcup.com/dbms/architecture-of-database.htm [WWW]

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Hanna Kristin Ojaveer

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Töögraafiku liidestuse loomine ja dokumentatsiooni kaasajastamine personalitarkvara Persona näitel “ , mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

[pp.kk.aaaa]

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Andmebaasist töögraafiku kirje informatsiooni päring

```
public class GetToograafikForApiMethod
{
  private readonly string _asutus;
  private readonly long _id;

  public GetToograafikForApiMethod(string asutus, long id) :
base("toograafik.get-toograafik-for-api")
  {
    _asutus = asutus;
    _id = id;
  }

  ...

   protected override ToograafikDto Execute()
   {
     var result =
       (from toograafik in ObjectContext.TgToograafikud
       where toograafik.Asutus == _asutus
             && toograafik.Id == _id
       select new ToograafikDto
       {
          Id = toograafik.Id,
          Algus = toograafik.Algus,
          Lopp = toograafik.Lopp,
          ToosuheId = toograafik.TsToosuheId,
          ToopostId = toograafik.TgToopostId,
          Kestus = toograafik.Kestus,
          Markus = toograafik.Markus,
          MuutAeg = toograafik.MuutAeg,
          PuhkusId = toograafik.TsPuhkusId,
          PuhkusLiikKl = new KlVaartusDto()
          {
            KlassifikaatorKood = TsPuhkusLiik.KlKood,
            KoodWithLisaId = toograafik.PuhkusLiikKl
           }
        }). FirstOrDefault();

      var asutusPuhkusLiikKls =
KlVaartusServices.GetKlVaartused(TsPuhkusLiik.KlKood
, null, false).ToList();

       if (result != null)
       {
         result.PuhkusLiikKl =
KlVaartusServices.InflateKlVaartusDto(result.PuhkusLiikKl,
asutusPuhkusLiikKls);
       }
       return result;
    }
 }
```

Lisa 3 – Andmebaasist töögraafiku kirje kustutamine

```
class DeleteToograafikApiMethod
{
    private readonly string _asutus;
    private readonly long _id;

    public DeleteToograafikApiMethod(string asutus, long id) :
base("toograafik.api-delete")
    {
        _asutus = asutus;
        _id = id;
    }

    ...

    protected override VoidResult Execute()
    {
        var toograafik = ObjectContext.TgToograafik
            .Where(t => t.Asutus == _asutus)
            .FirstOrDefault(t => t.Id == _id);

        Assert(() => toograafik != null, "toografikObjekt.not-found");

        if (toograafik.IsToopost)
        {
            var isToograafikKinnitatud = ExecuteMethod(new
IsToograafikKinnitatudMethod(_asutus, toograafik));
            Assert(() => !isToograafikKinnitatud, "related-toograafik-
kirje-has-already-been-approved", toograafik.DisplayValue);
        }

        var tooaeg = ObjectContext.TgTooajad
            .Where(t => t.Asutus == _asutus)
            .FirstOrDefault(t => t.TgToograafikId == toograafik.Id);

        if (tooaeg != null)
        {
            tooaeg.TgToograafikId = null;
            tooaeg.OnKontrollitud = false;
            tooaeg.PlaneeritudAlgus = null;
            tooaeg.PlaneeritudLopp = null;
            ObjectContext.TgTooajad.ApplyChanges(tooaeg);
        }

        ObjectContext.TgToograafik.DeleteObject(toograafik);

        return null;
    }
}
```

Lisa 4 – Loodud rakenduse kood

```
public class TookraafikudController : BaseApiController
{
    internal const string DateFormat = "yyyy-MM-dd";
    private static readonly Logger =
LogManager.GetCurrentClassLogger();

    [HttpPost]
    public TookraafikDto LisaTookraafik([FromBody] ModifyTookraafikDto
createTookraafik)
    {
        var requestBody = GetRequestBody();
        Logger.Info("LisaTookraafik");
        Logger.Info(requestBody);

        var asutusPuhkusLiikKls =
KlVaartusServices.GetKlVaartused(TsPuhkusLiik.KlKood, null,
false).ToList();
        UserRoleCheck(SecurityRole.ApiTookraafikudMuutmine);
        SisendValidation(createTookraafik);
        UserAccibleToosuheIdsCheck(createTookraafik);
        if (createTookraafik.PuhkusLiikKl != null)
        {
            CheckPuhkusliikKl(createTookraafik, asutusPuhkusLiikKls);
        }

        var user = PersonaUser.Current;
        var uusTookraafik = new TgTookraafik
        {
            Asutus = user.Asutus,
            Algus = createTookraafik.Algus,
            Lopp = createTookraafik.Lopp,
            Kestus = createTookraafik.Kestus,
            Markus = createTookraafik.Markus,
            TsToosuheId = createTookraafik.ToosuheId,
            TgToopostId = createTookraafik.ToopostId,
            PuhkusLiikKl = createTookraafik.PuhkusLiikKl
        };

        if (uusTookraafik.TgToopostId != null)
        {
            if
(GraafikudServices.IsTookraafikKinnitatud(uusTookraafik))
            {
                throw CreateHttpResponseException("Toiming
ebaõnnestus. Antud kuu töögraafik on juba kinnitatud. Lisamiseks tuleb
Töögraafikult vastava kuu kinnitus eemaldada.",
HttpStatusCode.BadRequest);
            }
        }
```

```
        try
        {
            GraafikudServices.SaveTootajateGraafikud(new[] {
uusToograafik }, new List<string>());

            DateTime arvestusKuu = new
DateTime(uusToograafik.Algus.Year, uusToograafik.Algus.Month, 1);
            GraafikudServices.ArvutaNormidCache(new List<long> {
uusToograafik.TsToosuheId }, arvestusKuu);

            var toograafikModel = new ToograafikDto
            {
                Id = uusToograafik.Id,
                Algus = uusToograafik.Algus,
                Lopp = uusToograafik.Lopp,
                ToosuheId = uusToograafik.TsToosuheId,
                ToopostId = uusToograafik.TgToopostId,
                Kestus = uusToograafik.Kestus,
                Markus = uusToograafik.Markus,
                MuutAeg = uusToograafik.MuutAeg,
                PuhkusId = uusToograafik.TsPuhkusId,
                PuhkusLiikKl = new KlVaartusDto()
                {
                    KlassifikaatorKood = TsPuhkusLiik.KlKood,
                    KoodWithLisaId = uusToograafik.PuhkusLiikKl
                }
            };
            toograafikModel.PuhkusLiikKl =
KlVaartusServices.InflateKlVaartusDto(toograafikModel.PuhkusLiikKl,
asutusPuhkusLiikKls);
            return toograafikModel;
        }
        catch (ServiceException se)
        {
            Logger.Error(se);
            throw HandleServiceException(se);
        }
        catch (Exception e)
        {
            Logger.Error(e);
            throw CreateHttpResponseException();
        }
    }

    [HttpPut]
    public ToograafikDto UpdateToograafik([FromUri] long id,
[FromBody] ModifyToograafikDto updateToograafik)
    {
        var requestBody = GetRequestBody();
        Logger.Info("UuendaToograafik");
        Logger.Info(requestBody);

        var asutusPuhkusLiikKls =
KlVaartusServices.GetKlVaartused(TsPuhkusLiik.KlKood, null,
```

```
false).ToList();

        UserRoleCheck(SecurityRole.ApiTöögraafikudMuutmine);
        SisendValidation(updateTöögraafik);
        UserAccibleToosuheIdsCheck(updateTöögraafik);
        if (updateTöögraafik.PuhkusLiikKl != null)
        {
            CheckPuhkusliikKl(updateTöögraafik, asutusPuhkusLiikKls);
        }

        var töögraafikFromDb =
GraafikudServices.GetTöögraafikForModification(id);
        if (töögraafikFromDb == null)
        {
            throw CreateHttpResponseException("Töögraafikut ei
ekstisteeri või pole Teie kasutajale antud selle nägemisõigust.",
HttpStatusCode.BadRequest);
        }
        if
(GraafikudServices.IsTöögraafikKinnitatud(töögraafikFromDb))
        {
            throw CreateHttpResponseException("Toiming ebaõnnestus.
Antud kuu töögraafik on juba kinnitatud. Uuendamiseks tuleb
Töögraafikult vastava kuu kinnitus eemaldada.",
HttpStatusCode.BadRequest);
        }

        töögraafikFromDb.Algus = updateTöögraafik.Algus;
        töögraafikFromDb.Lopp = updateTöögraafik.Lopp;
        töögraafikFromDb.Kestus = updateTöögraafik.Kestus;
        töögraafikFromDb.Markus = updateTöögraafik.Markus;
        töögraafikFromDb.TsToosuheId = updateTöögraafik.ToosuheId;
        töögraafikFromDb.TgToopostId = updateTöögraafik.ToopostId;
        töögraafikFromDb.PuhkusLiikKl = updateTöögraafik.PuhkusLiikKl;

        try
        {
            GraafikudServices.SaveTootajateGraafikud(new[] {
töögraafikFromDb }, new List<string>());
            var töögraafikModel = new TöögraafikDto
            {
                Id = töögraafikFromDb.Id,
                Algus = töögraafikFromDb.Algus,
                Lopp = töögraafikFromDb.Lopp,
                ToosuheId = töögraafikFromDb.TsToosuheId,
                ToopostId = töögraafikFromDb.TgToopostId,
                Kestus = töögraafikFromDb.Kestus,
                Markus = töögraafikFromDb.Markus,
                MuutAeg = töögraafikFromDb.MuutAeg,
                PuhkusId = töögraafikFromDb.TsPuhkusId,
                PuhkusLiikKl = new KlVaartusDto()
                {
                    KlassifikaatorKood = TsPuhkusLiik.KlKood,
                    KoodWithLisaId = töögraafikFromDb.PuhkusLiikKl
```

```
                }
            };
            toograafikModel.PuhkusLiikKl =
KlVaartusServices.InflateKlVaartusDto(toograafikModel.PuhkusLiikKl,
asutusPuhkusLiikKls);
            return toograafikModel;
        }
        catch (ServiceException se)
        {
            Logger.Error(se);
            throw HandleServiceException(se);
        }
        catch (Exception e)
        {
            Logger.Error(e);
            throw CreateHttpResponseException();
        }
    }

    [HttpGet]
    public GetAllToograafikudResponse GetAllToograafikud()
    {
        UserRoleCheck(SecurityRole.ApiToograafikudVaatamine);

        var paramsNvc =
HttpUtility.ParseQueryString(Request.RequestUri.Query);
        var toosuheIds = TryGetParamAsIdsList(Parameters.ToosuheId,
paramsNvc);
        var alates = TryGetParamAsDate(Parameters.Alates, paramsNvc,
MinDate);
        var kuni = TryGetParamAsDate(Parameters.Kuni, paramsNvc,
MaxDate);
        var alatesKuniSubjekt =
TryGetParamAsString(Parameters.AlatesKuniSubjekt, paramsNvc, "Algus");
        var limiit = TryGetParamAsNullableInt(Parameters.Limiit,
paramsNvc, 1000);
        var jarjestus = TryGetParamAsString(Parameters.Jarjestus,
paramsNvc, "Asc");
        var jarjestusSubjekt =
TryGetParamAsString(Parameters.JarjestusSubjekt, paramsNvc, "Algus");
        var nihe = TryGetParamAsNullableInt(Parameters.Nihe,
paramsNvc, 0);

        var userAccessibleToosuheIds =
ToosuhtedServices.GetUserAccessibleToosuhtedIds();
        var filterToosuheIdsAreValid = !toosuheIds.Any() ||
toosuheIds.Intersect(userAccessibleToosuheIds).Count() ==
toosuheIds.Count;
        if (!filterToosuheIdsAreValid)
        {
            var invalidToosuheIds =
toosuheIds.Except(userAccessibleToosuheIds).ToList();
            throw CreateHttpResponseException(
                $"Parameeter '{Parameters.ToosuheId}' => Järgnevaid
```

```
töösuhteid ei eksisteeri või pole Teie kasutajale antud nende
nägemisõigust: [{string.Join(ParamArrayDelimiter.ToString(),
invalidToosuheIds)}]",
                HttpStatusCode.BadRequest);
        }

        var isValidAlatesKuniSubjekt =
Enum.TryParse(alatesKuniSubjekt, ignoreCase: true, out
AlatesKuniSubjekt eAlatesKuniSubjekt);
        if (!isValidAlatesKuniSubjekt)
        {
            throw CreateHttpResponseException("Parameetri väärtus peab
olema 'Algus' või 'MuutAeg'.", HttpStatusCode.BadRequest);
        }
        var isValidJarjestus = Enum.TryParse(jarjestus,
ignoreCase:true ,out Jarjestus eJarjestus);
        if (!isValidJarjestus)
        {
            throw CreateHttpResponseException("Parameetri väärtus peab
olema 'Asc' või 'Desc'.", HttpStatusCode.BadRequest);
        }
        var isValidJarjestusSubjekt = Enum.TryParse(jarjestusSubjekt,
ignoreCase: true, out JarjestusSubjekt eJarjestusSubjekt);
        if (!isValidJarjestusSubjekt)
        {
            throw CreateHttpResponseException("Parameetri väärtus peab
olema 'Algus' või 'MuutAeg'.", HttpStatusCode.BadRequest);
        }
        if (alates > kuni)
        {
            throw
CreateHttpResponseException(ErrorMessages.ParamAlatesCannotBeGreaterTh
anParamKuni, HttpStatusCode.BadRequest);
        }
        if (limiit > 1000)
        {
            throw
CreateHttpResponseException(ErrorMessages.ParamLimiitIsTooBig,
HttpStatusCode.BadRequest);
        }

        try
        {
            var result =
GraafikudServices.GetToograafikudForApi(alates, kuni,
eAlatesKuniSubjekt, limiit, eJarjestus, eJarjestusSubjekt, nihe,
toosuheIds);
            var response = new GetAllToograafikudResponse
            {
                Parameeterid = ParametersData(toosuheIds, alates,
kuni, alatesKuniSubjekt, limiit, jarjestus, jarjestusSubjekt, nihe,
result.TotalResult),
                Tulemus = result.Result,
                Teated = NotificationsList( nihe, result.TotalResult,
```

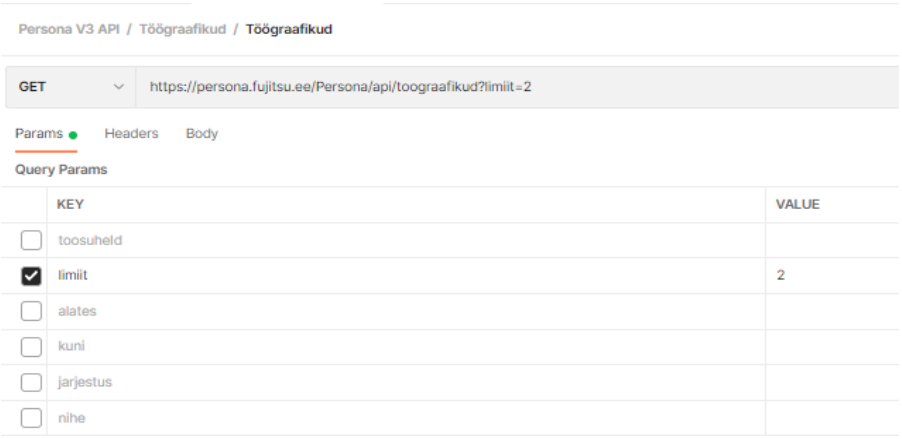
```
limiit)
            };
            return response;
        }
        catch (ServiceException se)
        {
            Logger.Error(se);
            throw HandleServiceException(se);
        }
        catch (Exception e)
        {
            Logger.Error(e);
            throw CreateHttpResponseException();
        }
    }

    [HttpGet]
    public ToograafikDto GetToograafik(long id)
    {
        UserRoleCheck(SecurityRole.ApiToograafikudVaatamine);

        try
        {
            var toograafik =
GraafikudServices.GetToograafikForApi(id);
            if (toograafik == null)
                throw
CreateHttpResponseException(ErrorMessages.SisendiViga,
                    HttpStatusCode.NotFound);
            return toograafik;
        }
        catch (HttpResponseException)
        {
            throw;
        }
        catch (ServiceException se)
        {
            Logger.Error(se);
            throw HandleServiceException(se);
        }
        catch (Exception e)
        {
            Logger.Error(e);
            throw CreateHttpResponseException();
        }
    }

    [HttpDelete]
    public DeleteResponse DeleteToograafik(long id)
    {
        UserRoleCheck(SecurityRole.ApiToograafikudMuutmine);

        try
        {
```

```
            GraafikudServices.DeleteToograafikApi(id);
            return new DeleteResponse();
        }
        catch (ServiceException se)
        {
            Logger.Error(se);
            throw HandleServiceException(se);
        }
        catch (Exception e)
        {
            Logger.Error(e);
            throw CreateHttpResponseException();
        }
    }

    private static List<string> ParametersData(List<long> toosuheIds,
DateTime alates, DateTime kuni, string alatesKuniSubjekt, int? limiit,
string jarjestus, string jarjestusSubjekt, int? nihe, int totalResult)
    {
        var result = new List<string>();

        if (toosuheIds.Any())
        {
            result.Add("Valitud töösuhted: " + string.Join(",",
toosuheIds));
        }
        result.Add("Tulemusi kokku: " + totalResult);
        result.Add("Määratud limiit: " + limiit);
        result.Add("Alates kuupäev: " + alates.ToString(DateFormat));
        result.Add("Kuni kuupäev: " + kuni.ToString(DateFormat));
        result.Add("Alates-kuni subjekt: " + alatesKuniSubjekt);
        result.Add("Määratud järjestus: " + jarjestus);
        result.Add("Määratud järjestuse subjekt: " +
jarjestusSubjekt);
        result.Add("Määratud nihe: " + nihe);

        return result;
    }

    private static List<string> NotificationsList(int? nihe, int
totalResult, int? limiit)
    {
        var notifications = new List<string>();
        if (nihe >= totalResult)
        {
            notifications.Add("Määratud nihe on suurem kui tulemusi
kokku.");
        }

        if (limiit == 0)
        {
            notifications.Add("Määratud limiit on 0. Tulemusi ei
kuvata");
        }
```

```
        if (totalResult == 0)
        {
            notifications.Add("Tulemusi ei leitud. Töögraafikuid ei
eksisteeri või pole Teie kasutajale antud nende nägemisõigust.");
        }

        return notifications;
    }

    private static void UserAccibleToosuheIdsCheck(ModifyToograafikDto
createToograafik)
    {
        var userAccessibleToosuheIds = GetUserAccessibleToosuheIds();
        var toosuheIdValid =
userAccessibleToosuheIds.Contains(createToograafik.ToosuheId);
        if (!toosuheIdValid)
        {
            throw CreateHttpResponseException("Töösuhet ei ekstisteri
või pole Teie kasutajale antud nende nägemisõigust.");
        }
    }

    private static void SisendValidation(ModifyToograafikDto
toograafik)
    {
        var algus = toograafik.Algus;
        var lopp = toograafik.Lopp;
        var kestus = toograafik.Kestus;

        if (DateTime.Compare(DateTime.MinValue, algus) == 0)
            throw CreateHttpResponseException($"Sellist 'Algus'
kuupäeva ei eksisteeri.", HttpStatusCode.BadRequest);
        if (DateTime.Compare(DateTime.MinValue, lopp) == 0)
            throw CreateHttpResponseException($"Sellist 'Lõpp'
kuupäeva ei eksisteeri.", HttpStatusCode.BadRequest);

        if (!IsDateTimeValid(algus))
            throw CreateHttpResponseException($"'{nameof(algus)}' peab
olema vahemikus '{MinDate.ToString(DateFormatIso8601)}' kuni
'{MaxDate.ToString(DateFormatIso8601)}'.", HttpStatusCode.BadRequest);
        if (!IsDateTimeValid(lopp))
            throw CreateHttpResponseException($"'{nameof(lopp)}' peab
olema vahemikus '{MinDate.ToString(DateFormatIso8601)}' kuni
'{MaxDate.ToString(DateFormatIso8601)}'.", HttpStatusCode.BadRequest);

        if (algus > lopp)
            throw CreateHttpResponseException($"'Algus' ei saa olla
hilisem kui 'Lõpp'.", HttpStatusCode.BadRequest);
        if (algus == lopp)
            throw CreateHttpResponseException($"Algus- ja lõppaeg
kattuvad.", HttpStatusCode.BadRequest);

        if (kestus < 0)
```

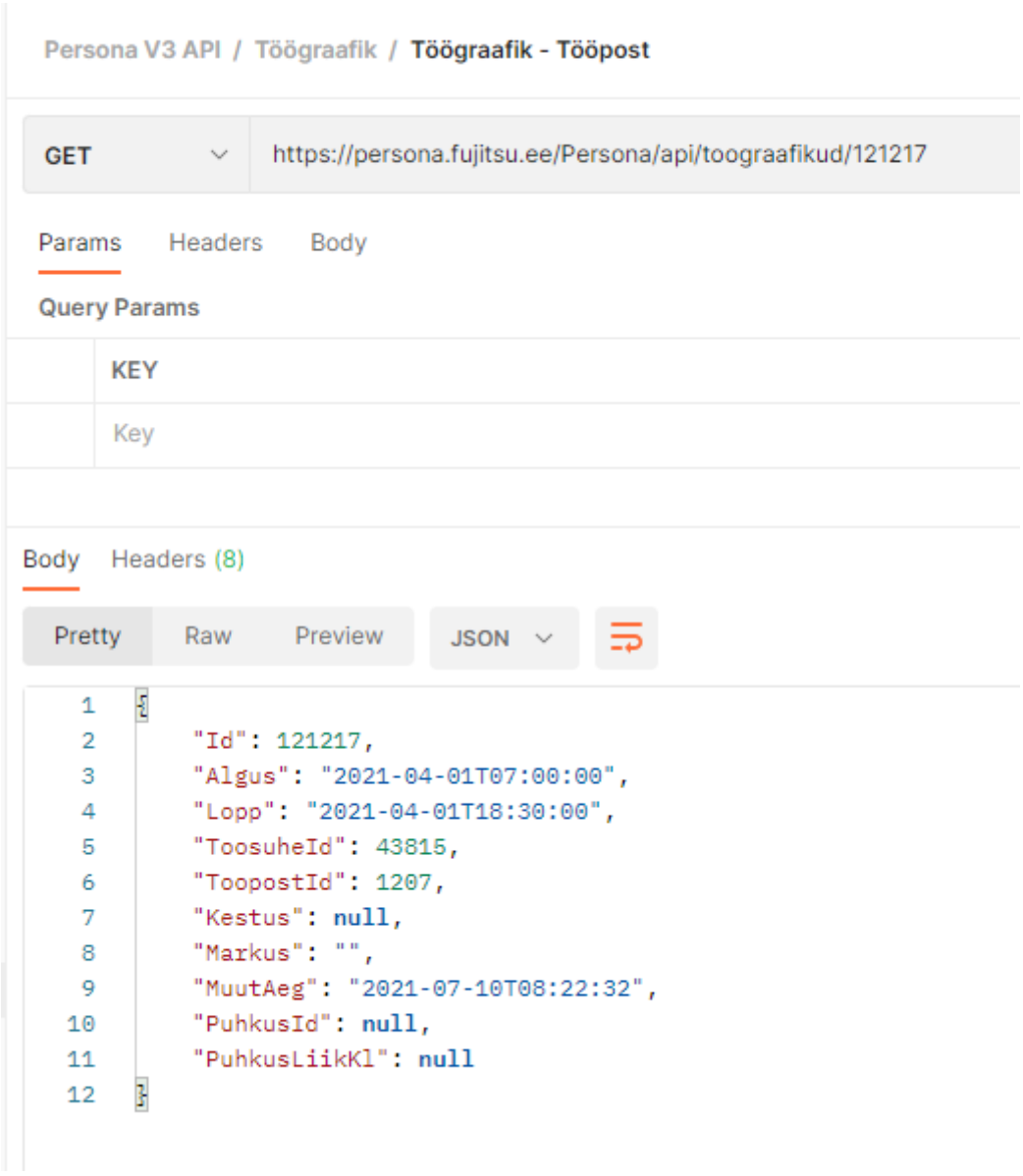
```
            throw CreateHttpResponseException($"'Kestus' väärtus peab
olema suurem või võrdne nulliga", HttpStatusCode.BadRequest);

        if (toograafik.ToosuheId == 0)
            throw CreateHttpResponseException("Töösuhe on
väärtustamata", HttpStatusCode.BadRequest);
        if (toograafik.ToopostId != null && toograafik.PuhkusLiikKl !=
null)
            throw CreateHttpResponseException("Väljad 'ToopostId' ja
'PuhkusLiikKl' ei tohi olla korraga määratud.",
HttpStatusCode.BadRequest);
    }

    private static void UserRoleCheck(SecurityRole userRole)
    {
        var user = PersonaUser.Current;
        if (!user.IsInRole(userRole))
            throw CreateHttpResponseException(
                "Kasutajal puuduvad vastavad õigused päringu
sooritamiseks.",
                HttpStatusCode.Forbidden);
    }

    private static void CheckPuhkusliikKl(ModifyToograafikDto
toograafik, List<KlVaartusEntity> klVaartusEntities)
    {
        var checkPuhkusLiikKl = new KlVaartusDto()
        {
            KlassifikaatorKood = TsPuhkusLiik.KlKood,
            KoodWithLisaId = toograafik.PuhkusLiikKl
        };

        if (KlVaartusServices.InflateKlVaartusDto(checkPuhkusLiikKl,
klVaartusEntities) == null)
            throw CreateHttpResponseException("Sisestatud PuhkusLiikKl
on vigane.", HttpStatusCode.BadRequest);
    }
}

public class GetAllToograafikudResponse
{
    public List<string> Parameeterid { get; set; }
    public List<string> Teated { get; set; }
    public List<ToograafikDto> Tulemus { get; set; }
}

public class DeleteResponse
{
    public string Tulemus { get; set; } = "Kustutamine õnnestus";
}
```

Lisa 5 – *GET* töögraafikute kirjete päringu näidis

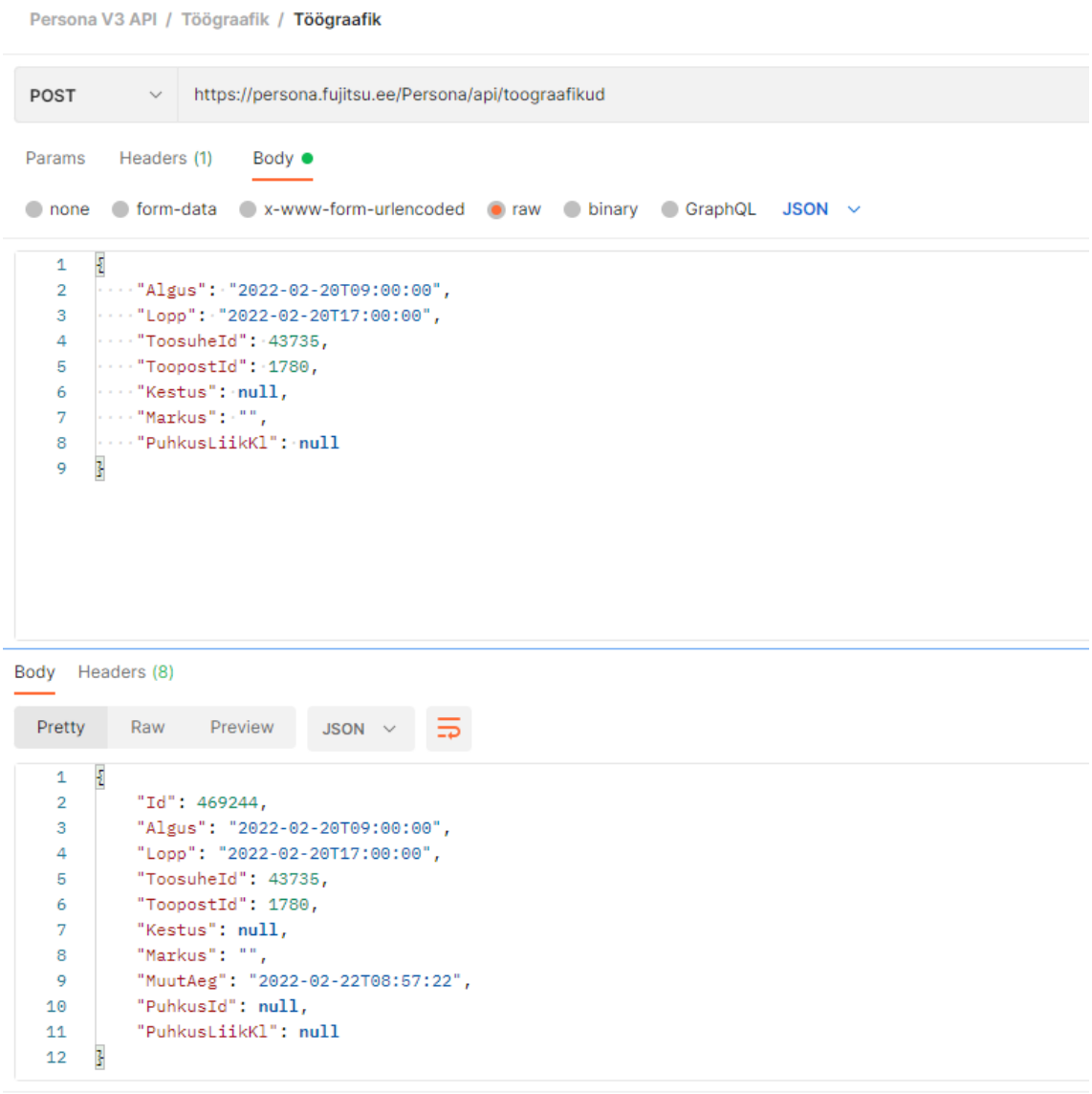



```
{
    "Parameeterid": [
        "Tulemusi kokku: 10",
        "Määratud limiit: 2",
        "Alates kuupäev: 1900-01-01",
        "Kuni kuupäev: 2100-12-31",
        "Määratud järjestus: Asc",
        "Määratud nihe: 0"
    ],
    "Teated": [],
    "Tulemus": [
        {
            "Id": 466354,
            "Algus": "2021-05-03T07:30:00",
            "Lopp": "2021-05-03T16:00:00",
            "ToosuheId": 43830,
            "ToopostId": 1780,
            "Kestus": null,
            "Markus": "",
            "MuutAeg": "2021-08-06T11:48:35",
            "PuhkusId": null,
            "PuhkusLiikKl": null
        },
        {
            "Id": 463956,
            "Algus": "2021-12-22T00:00:00",
            "Lopp": "2021-12-30T00:00:00",
            "ToosuheId": 43740,
            "ToopostId": null,
            "Kestus": null,
            "Markus": "",
            "MuutAeg": "2021-07-06T11:15:09",
            "PuhkusId": 104256,
            "PuhkusLiikKl": {
                "UnikaalneKood": "TsPuhkusLiik|Pohipuhkus|1",
                "Nimetus": "Põhipuhkus 28"
            }
        }
    ]
}
```

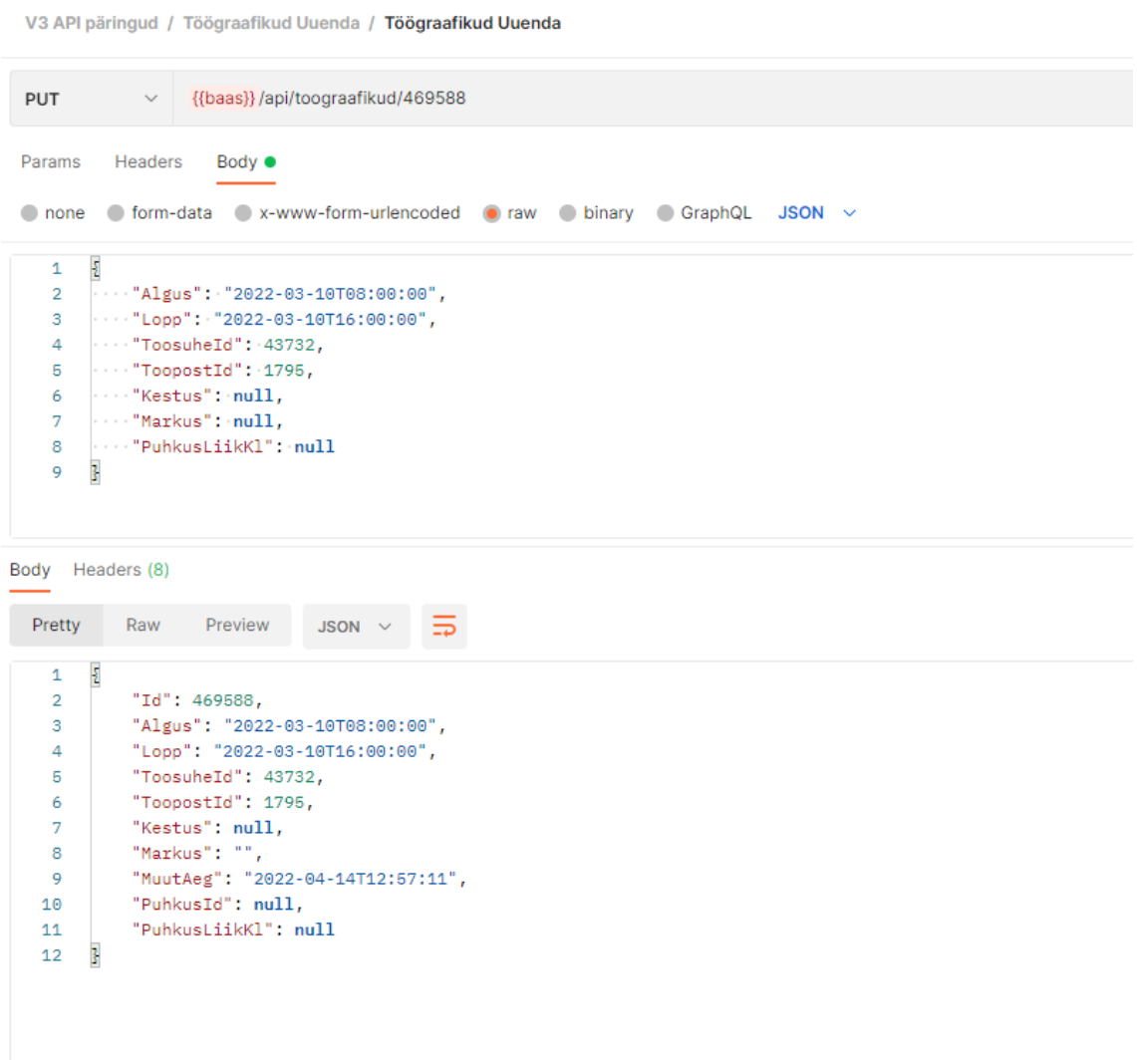
Lisa 6 – *GET* töögraafiku kirje päringu näidis



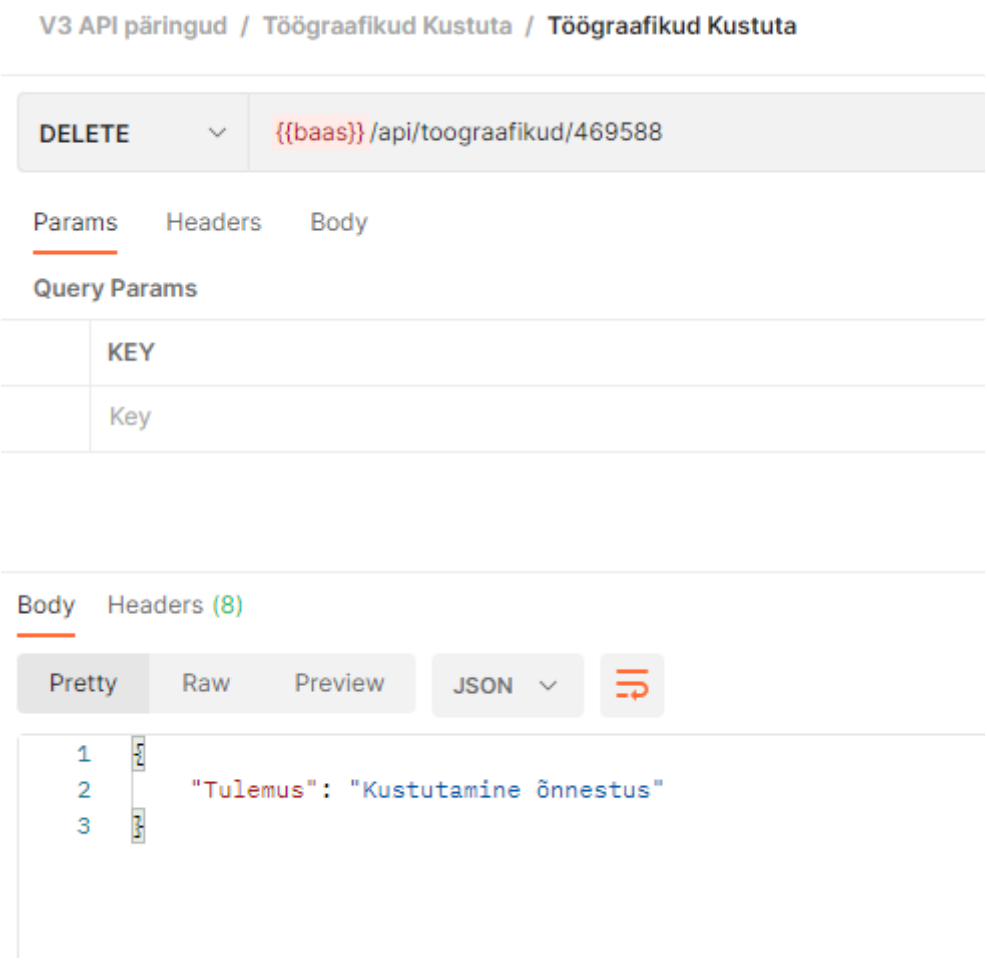
Lisa 7 – *POST* töögraafiku kirje päringu näidis



Lisa 8 – *PUT* töögraafiku kirje päringu näidis



Lisa 9 – *DELETE* töögraafiku kirje päringu näidis



Lisa 10 – Dokumentatsioon