TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Arvutiteaduse instituut
Informaatika õppekava

Elisabeth Räni

Changelog API – veebipõhine muudatuste logimise teenus

Bakalaureusetöö (9 EAP)

Juhendajad: Erik Räni, MSc
Ljubov Jaanuska, PhD

Tartu 2025

Changelog API – veebipõhine muudatuste logimise teenus

Lühikokkuvõte:

Aja jooksul avaldavad arendajad tarkvaraprojektidest uusi versioone, millesse kuulub üks või mitu muudatust. Tihti kasutavad nad muudatuste info talletamiseks tekstifaili, veebilehte, jaotusplatvormi tutvustuslehte, mõnikord ka tarkvarasisest spetsiaalset vaadet. Projektist huvitatud inimesed saavad tänu nendele allikatele teha otsuse, millist versiooni nad soovivad kasutada. Käesoleva bakalaureusetöö käigus arendatakse avatud lähtekoodiga veebipõhine teenus Changelog API, mis viib muudatuste logi halduse ühte kesksesse kohta ja kehtestab selle sisule ühtsed reeglid.

Võtmesõnad:
Muudatuste logi, väljalaskemärkmed, API (rakenduse programmeerimisliides), veebiteenus, versioonikontroll, versiooniajalugu, dokumentatsioon, semantiline versioonide loomine

CERCS: P175 Informaatika, süsteemiteooria

Changelog API – web-based change logging service

Abstract:

Over time, developers release new versions of software projects that include one or more changes. Information regarding these changes is typically documented in text files, web pages, distribution platform introduction pages, or dedicated sections within the software itself. Users and stakeholders rely on these sources to assess version updates and determine the most suitable release for their needs. This bachelor's thesis aims to develop an open-source, web-based service that centralizes changelog management and establishes standardized guidelines for its content.

Keywords:
Changelog, release notes, API (Application Programming Interface), web service, version control, version history, documention, semantic versioning

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Tarkvara väljalaskeid toimub peaaegu iga päev, mistõttu muudatuste kirjapanek on oluline. Kirjalik dokumentatsioon aitab mõista arengut ja jälgida tehtud muudatusi. Iga väljalase üldiselt dokumenteeritakse, kuid mõnikord esineb nendes kirjapanekutes puudusi, mis võivad tuleneda ühtsete reeglite puudumisest. Seetõttu on soovitatav kehtestada selged juhised muudatuste logimiseks.

Käesoleva töö eesmärk on luua veebirakendus Changelog API, mis suunab kasutajaid järgima muudatuste logimise häid tavasid: semantiline versioonimine, muudatuste klassifitseerimine, autori mainimine, info grupeerimine ja muudatuste kategoriseerimine. Administreerimiseks on mõeldud HTTP (ingl *Hypertext Transfer Protocol*) API (ingl *Application Programming Interface*) ning tavakasutajatele ühe projekti muudatuste logi vaatamiseks tavaline HTML (ingl *Hypertext Markup Language*) veebileht. API lõpp-punktide loomisel peetakse silmas võimalikke veaolukordi, et kuvada kasulikke kasutajasõbralikke veateateid. Veebirakenduse arendatakse programmeerimiskeeles Python kasutades mikroraamistikku Flask. Valminud versioonide haldamise tarkvara on kättesaadav aadressil `https://changelogapi.eu`.

Töö on jaotatud kolmeks peamiseks peatükiks. Esimeses tausta peatükis analüüsitakse teadusartikleid ja veebilehti. Nendest leitakse versioonide logimise häid tavasid, mida kasutatakse vaba lähtekoodiga tarkvara (ingl *open-source software*) väljalasete võrdluse tingimustena. Võrdlemiseks valiti 16 avatud lähtekoodiga rakendust: React, Vue.js, Angular, Flask, Node.js, MySQL, Nginx, Next.js, MongoDB, PostgreSQL, Homebrew, Ansible, Apache httpd, Docker engine, Prometheus ning Kubernetes. Ühtlasi saadakse ülevaade tööriistadest, mida praegu kasutatakse ning mida Changelog API asendama hakkab.

Tarkvara arendusest ja analüüsist kirjutatakse kahes eraldi peatükis. Programmeerimise skoopi aitavad kindlates piirides hoida välja selgitatud kasutajarollid, funktsionaalsed ja mittefunktsionaalsed nõuded ning põhiline äriprotsess. Kasutajatüübid tuvastati eesmärkidest lähtuvalt: tarkvara autorid soovivad muudatuste logi sisu luua, samas tarkvara kasutajad tahavad seda sisu tarbida. Arenduse alla kuulub ka süsteemiadministreerimine Linuxi serveris. Pööratakse tähelepanu protsesside keskkonna eraldamisele läbi Incuse süsteemikonteinerite.

1 Versioonimine ja muudatuste logimine

Käesolev peatükk annab ülevaate versioonimisest ja muudatuste logimisest. Esimeses alapeatükis tutvutakse teadusartiklite ja veebiressurssidega, mis on headest muudatuste logimise ning semantilise versioonimise praktikatest. Seejärel võrreldakse avatud lähtekoodiga rakenduste väljalaskeid eelnevalt leitud heade tavade abil. Viimaks tuuakse välja, milliseid olemasolevaid lahendusi Changelog API asendama saaks hakata.

1.1 Muudatuste logid

Tarkvaraarendus on pidev ja dünaamiline protsess, kus uusi versioone ja täiustusi tehakse regulaarselt. Selleks, et kasutajad ja arendajad saaksid muudatustest selge ülevaate, dokumenteeritakse tarkvara arengu käigus tehtud muudatusi. Muudatuste logi (ingl *changelog*) ja tarkvara väljalaskemärkmed (ingl *release notes*) aitavad arendajatel paremini anda edasi infot tarkvara kasutajatele ja jälgida tarkvara arengut [1, 2]. Logid näitavad muudatusi, parandusi ja uusi funktsioone [1]. Väljalaske märkmete koostamine võib olla keerukas protsess, kus esineb sageli puudusi järgnevates aspektides [1]:

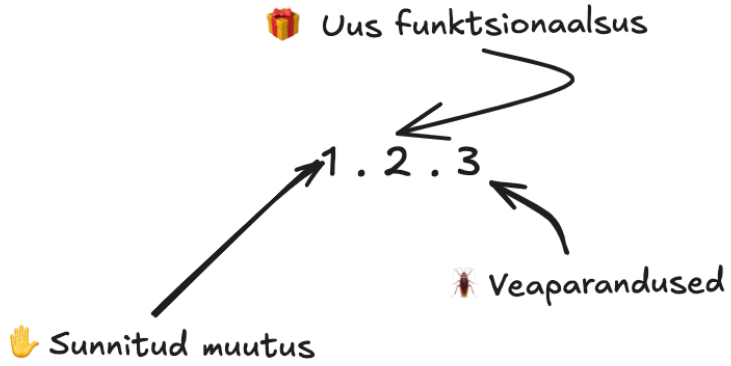
- sisu täielikkuses ja täpsuses,
- vormistuse stiilis ja grammatikas,
- märkmete viite aegumises ja raskesti leitavas asukohas,
- protsessi automatiseerimises, ette planeerimises ja ühtse stiili järgimises mitmes asukohas.

Näiteks macOS rakendustepoes App Store oleva Kindle'i tarkvara versioonist 7.6 alates esineb paljudel versioonidel täpselt ühesugune muudatuste kirje: „Several experience improvements and bug fixes.“ Ei täpsustata, mis tegelikult muutus. Automaatsete tööriistade, nagu Release Drafter[1], ja standardsete vormingute kasutamine aitab nimetatud probleeme leevendada, parandades versioonide logi kvaliteeti ja vähendades muudatuste logi koostamise aega [1]. Hea muudatuste logi sisaldab [2]:

1. iga tehtud uuendust,
2. põhjust,
3. asukohta,
4. autori nime,
5. autori kontaktandmeid,
6. kuupäeva.

[1] `https://github.com/release-drafter/release-drafter`

Semantiline versioonimine (ingl *semantic versioning*) on süsteem, mis määratleb, et versiooninumber koosneb kolmest numbrist (joonis 1): *MAJOR*, *MINOR*, *PATCH* [3]. Nendest esimese numbri suurendamine tähistab suurt muutust, mis sunnib tarkvarast sõltuvaid kasutajaid olemasolevaid töövooge muutma. Teise puhul ainult lisatakse uusi funktsionaalsuseid, mis tarkvara uuendamise korral ei too kaasa lisatööd. Kolmas, *PATCH*, tähistab veaparandusi.

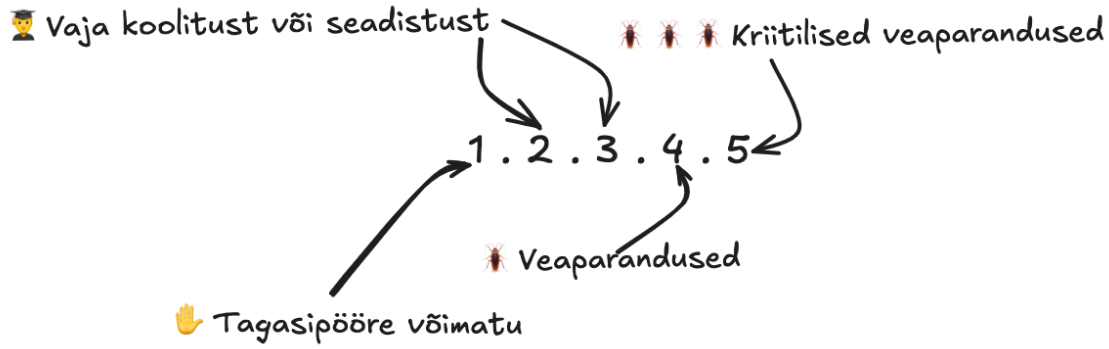


Joonis 1. Semantiline versioonimine kolme numbriga

Järgnev lõik tugineb Ahlbrandti artiklile [4]. Näiteks meditsiinilise tarkvara puhul on oluline tagada selge ja arusaadav dokumentatsioon, mis aitab kasutajatel mõista tarkvara muudatuste mõju nende tööle ja ohutusele. On pakutud välja, et versiooninumber võiks koosneda viiest punktidega ühendatud arvust (joonis 2). Esimene nendest suureneb siis, kui muudetud tarkvara variandi kasutuselevõtt tähendab, et eelnevale versioonile pole enam võimalik tagasi minna. Teine arv suureneb, kui on tarvis läbi viia märkimisväärne seadistuse muudatus või kasutajate koolitus. Kolmanda numbri tähendus on nagu teisel, aga väiksema olulisuse astmega. Neljanda puhul uut seadistust ega koolitust vaja pole – tähistatakse veaparandusi. Viimane number kasvab kriitiliste vigade korral.

Teises allikas „Keep a Changelog“ [5] jagatakse juhiseid inimestele lihtsasti loetava muudatuste logi kohta. Sealsete soovituste kohaselt ei tohiks puududa ükski tarkvaras toimunud muudatus. Kergemaks lugemiseks soovitatakse kasutada sama tüüpi informatsiooni grupeerimist järgnevatesse osadesse: lisatud, muudetud, soovimatuks kuulutatud, eemaldatud, parandatud ja turvaliseks muudetud funktsioonid. Need osad kuuluvad omakorda versioonidesse, mis on lingitavad ja sorteeritud väljalaske kuupäevade järgi alates hilisemast. Vaja on mainida, kas järgitakse semantilise versioonimise `semver.org` lehel esitatud reegleid.

Leitud heade tavade põhjal võrreldakse järgmises peatükis vaba lähtekoodiga tarkvara versioonimist. Kronoloogiline järjestatus alates hiliseimast versioonist ning nende ver-



Joonis 2. Viiest numbrist koosnev versiooninumber

sioonide lingitavus hõlbustavad muudatuste logi lugemist. Sama tüüpi info grupeerimine parendab loetavust. Semantiline versioonimine annab edasi, kas tarkvara kasutaja peab muutma oma olemasolevaid töövooge. Autori mainimine tagab tehtud töö tegijale tunnustuse, sest see tähtsustab inimese panust ning pingutusi. Tabelis 1 on kirjas võrdluses kasutatavad tavad ja nende kontrollimismeetodid.

Tabel 1. Versioonimise võrdlus

Hea tava	**Kontrollimismeetod**
Sorteeritud versioonid	Versioonid on sorteeritud väljalaske kuupäevade järgi alates hilisemast.
Info grupeerimine	Sama tüüpi muudatused paiknevad lähestikku.
Versioonide lingitavus	On võimalik salvestada link kindlale versiooninumbrile.
Semantiline versioonimine	Viimased viis versiooni vastavad semantilise versiooni reeglitele `semver.org`.
Autori mainimine	Igal muudatusel on juures autor.

1.2 Vaba lähtekoodiga tarkvara versioonimise võrdlus

Valiti välja 16 avatud lähtekoodiga rakendust (tabel 2, lisa 1), mille puhul rakendati tabelis 1 toodud metoodikat 2024. aasta detsembris. Valiti rakendused, millega oldi eelnevalt kokku puudutud või millest kuuldud. Kõikide vaadeldud rakenduste versioonid olid sorteeritud väljalaske kuupäevade järgi alates hilisemast. React, Next.js ja Node.js vastasid kõigile kriteeriumitele. PostgreSQLi versioonide info vastas ainult kahele. Tarkavara nginx keskkonna GitHub lehe versioonide väljalasete alamlehel oli rohkem headele tavadele vastavat informatsiooni, kuid leht viitas lisas 1 toodud asukohale kui ametlikule muudatuste logile, millest lähtuti. Tabeli 1 kriteeriumitest eirati enim autori mainimist.

Tabel 2. Versioonimise võrdlus

	React	**Vue.js**	**Angular**	**Flask**	**Node.js**	**MySQL**	**Nginx**	**Next.js**	**MongoDB**	**PostgreSQL**	**Homebrew**	**Ansible**	**Apache**	**Docker**	**Prometheus**	**Kubernetes**
sorteeritud versioonid	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
info grupeerimine	X	X	X		X	X	X	X				X	X	X	X	X
versioonide lingitavus	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X
semantiline versioonimine	X	X	X	X	X		X	X	X			X		X	X	X
autori mainimine	X				X			X			X					

Tabeli 2 põhjal on näha, et kõik vaadeldud avatud lähtekoodiga rakendused ühel või teisel viisil pakkusid avalikult lugemiseks muudatuste logi. Võib eeldada, et nende vastutavad hooldajad juba kasutavad tööriistu, mis aitavad luua muudatuste logisid. Selleks et logide kvaliteet oleks parem ja logid universaalsemad soovib antud bakalaureusetöö autor pakkuda oma lahendust Changelog API. Need samad eelpool mainitud inimesed võiksid olla tulevikus potentsiaalsed Changelog API kliendid, kelle veenmiseks peab loodav veebipõhine muudatuste logimise teenus olema selgelt parem praegustest. Järgmises alapeatükis kirjeldatakse mõningaid praegu kasutatavaid versioonimisega seotud töövooge ning hinnatakse, kui kerge on nendega häid tavasid rikkuda. Changelog API kavatsetakse luua nii, et oleks võimalikult lihtne parimaid praktikaid tabelist 1 järgida.

1.3 Mida Changelog API asendama hakkab?

Tekstipõhine lähenemine, nagu e-kirjade saatmine, *wiki* lehtede pidamine või *CHANGELOG.md* faili kasutamine, on lihtsam ja ligipääsetavam võimalus muudatuste dokumenteerimiseks. Samuti on rakenduste haldajatel võimalik lisada tekstilisi sõnumeid *git* siltidele nagu joonisel 3. Tekstipõhine viis ei sea piiranguid heade tavade rikkumiseks. Näiteks võib unustada info grupeerimise, ilma et miski selle olulisust meelde tuletaks.

Rakendustepõhine lähenemine suunab muudatuste logi koostajaid protsessi vältel järgima mõningaid häid tavasid. Tööriistad nagu GitHub Releases (joonis 4), GitLab Releases, Jira Releases ja `readme.com`-i *changelog* funktsionaalsus teevad versioonide väljalaske

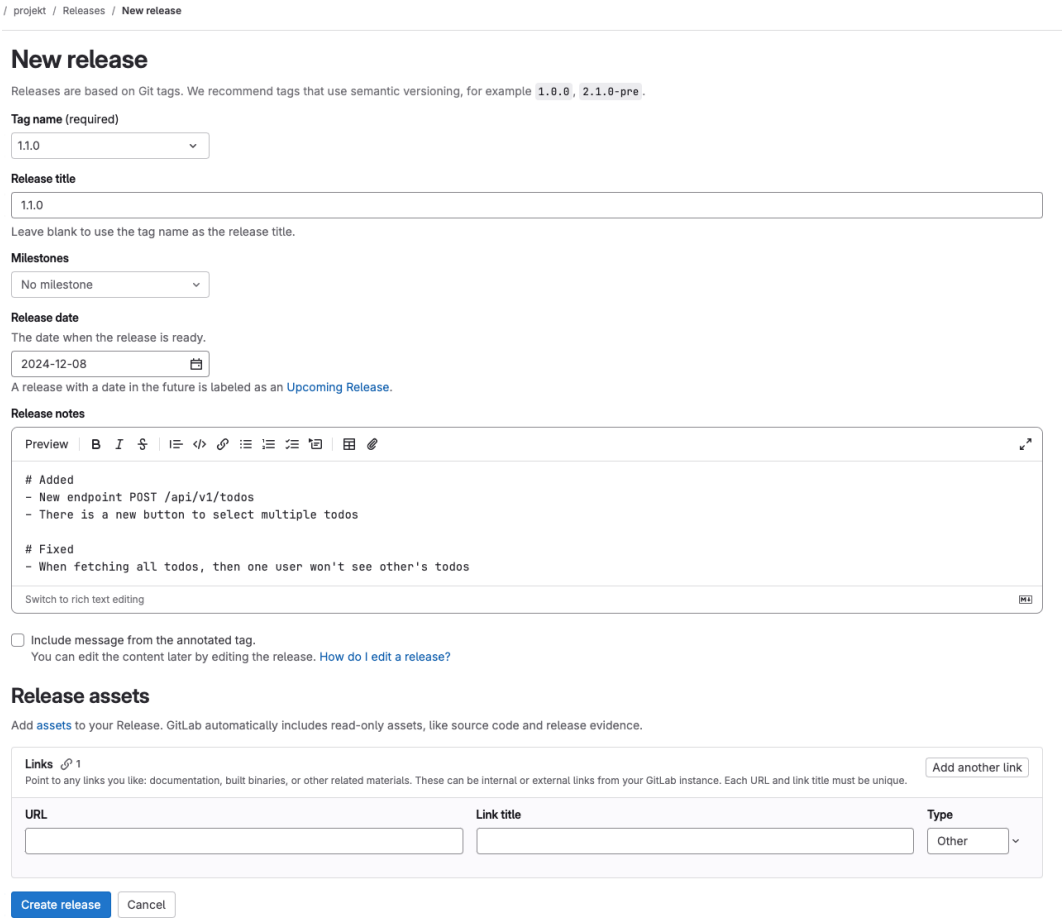
```
git tag -a 1.1.0 -m 'changes are
# Added
- New endpoint POST /api/v1/todos
- There is a new button to select multiple todos

# Fixed
- When fetching all todos, then one user won't see others todos'
```

Joonis 3. Versiooni muudatuste logi *git* sildi käsu kaudu

kuupäevade järgi sorteerimise lihtsaks. Samuti võimaldavad mainitud platvormid versioonide lingitavuse automaatselt. Samas, kuna põhiinfo esitatakse lihttekstina, ei seata piiranguid selle kohta, kuidas infot grupeeritakse või kuidas autoreid mainitakse. GitLabi näitel (joonis 4) soovitatakse küll semantilise versioonimise kasutamist, kuid selle järgimine ei ole kohustuslik.

Leidub ka tööriistu, mis versioonihaldussüsteemi *git commit*-ide põhjal genereerivad muudatuste logi. Näiteks tööriist *semantic-release*[2] eeldab, et tarkvaraarendajad järgivad kindlaid reegleid *commit*-ide sõnumites, et hiljem analüüsida sõnumite ajalugu ja kokku panna muudatuste logi. Sarnase tööpõhimõttega on *release-it*[3]. Eelnevalt mainitud lahendused võivad aidata järgida mõnda head tava, nagu näiteks info grupeerimine ja autori mainimine. Nende tööpõhimõte aga on vastuolus „Keep a Changelog“ lehe soovitusega mitte esitada *git commit* sõnumeid otse muudatuste logi kirjetena [5]. Soovitus on antud, sest iga *git commit* kirjeldab lähtekoodi muudatust, aga kasutaja jaoks on oluline rakenduse kasutamisega seotud muudatused [5]. Näiteks palju müra võivad tekitada dokumentatsiooni muutmisega seotud *git commit* sõnumid [5].

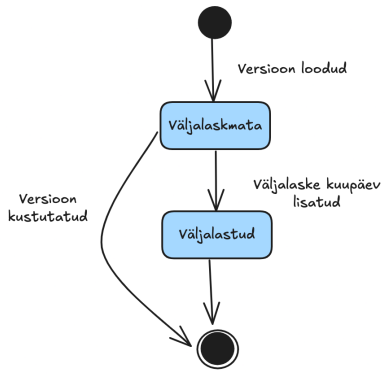
[2] https://semantic-release.gitbook.io/semantic-release
[3] https://github.com/release-it/release-it



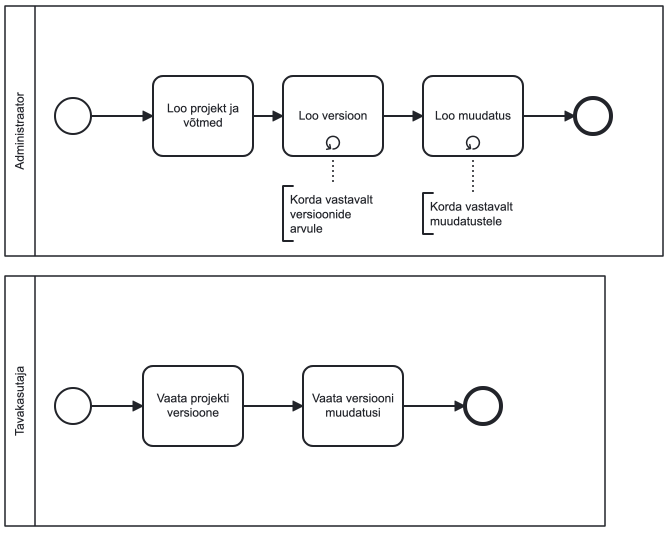
Joonis 4. Versiooni muudatuste logi GitLab Release'i kaudu

2 Tarkvara arenduse analüüs

Töö eesmärk on ehitada veebiteenus, mis motiveerib tarkvara loojaid järgima muudatuste logimise häid praktikaid. Käesolev peatükk keskendub tarkvara analüüsile, seades paika selged kasutajarollid, funktsionaalsed ja mittefunktsionaalsed nõuded ning rakenduse äriprotsessi. Changelog API on planeeritud olema veebipõhine teenus, millel on kaks osa: piiratud ligipääsuga API lõpp-punktid ja kõigile ligipääsetavad veebilehed. API lõpp-punktidega luuakse uusi versioone ning nendesse kuuluvaid muudatusi. Loodud sisu avaldatakse lihtsasti loetaval HTML-kujul. Iga versioon on kas avaldatud või avaldamata olekus (joonis 5). Administreerivaid kasutajaid pannakse järgima kõiki tabelis 1 esitatud häid tavasid.



Joonis 5. Versiooni avaldamise olekuskeem



Joonis 6. Äriprotsess

2.1 Kasutajad ja kasutuslood

Rakendusel on kaks kasutajat: administraator ja tavakasutaja. Viimase eesmärk on teada saada, mis on tarkvara juures muutnud (joonis 6). Esimene soovib võimalikult täpselt seda infot edasi anda. Kokku tuvastati kaheksa kasutuslugu (joonis 7). Enamik nendest sisaldab ainult ühte põhilise töövoo sammu, milleks on vastava API käsu kasutamine soovitud argumentidega. Järgnevalt on esitatud kaks põhilist kasutuslugu.

Kasutuslugu 1: Uue versiooni avaldamine

Tegutsejad: Administraator

Eeltingimus: Projekt ja versioon on loodud.

Põhiline töövoog:

1. Administraator loob versioonile muudatuskirjeid.
2. Administraator lisab versioonile väljalaske kuupäeva.

Lisatöövood:

1a. Muudatuse väärtus, tüüp või versiooni id on puudu.
    1. Süsteem vastab veateatega.
1b. Administraator otsustab, et muudatus ei kuulu sellesse versiooni.
    1. Administraator tõstab muudatuse uude versiooni.
1c. Administraator otsustab, et muudatus ei kuulu ühtegi versiooni.
    1. Administraator kustutab muudatuse ära.

Kasutuslugu 2: Versioonide lugemine

Tegutsejad: Tavakasutaja

Eeltingimus: Projekt on loodud koos versioonidega.

Põhiline töövoog:

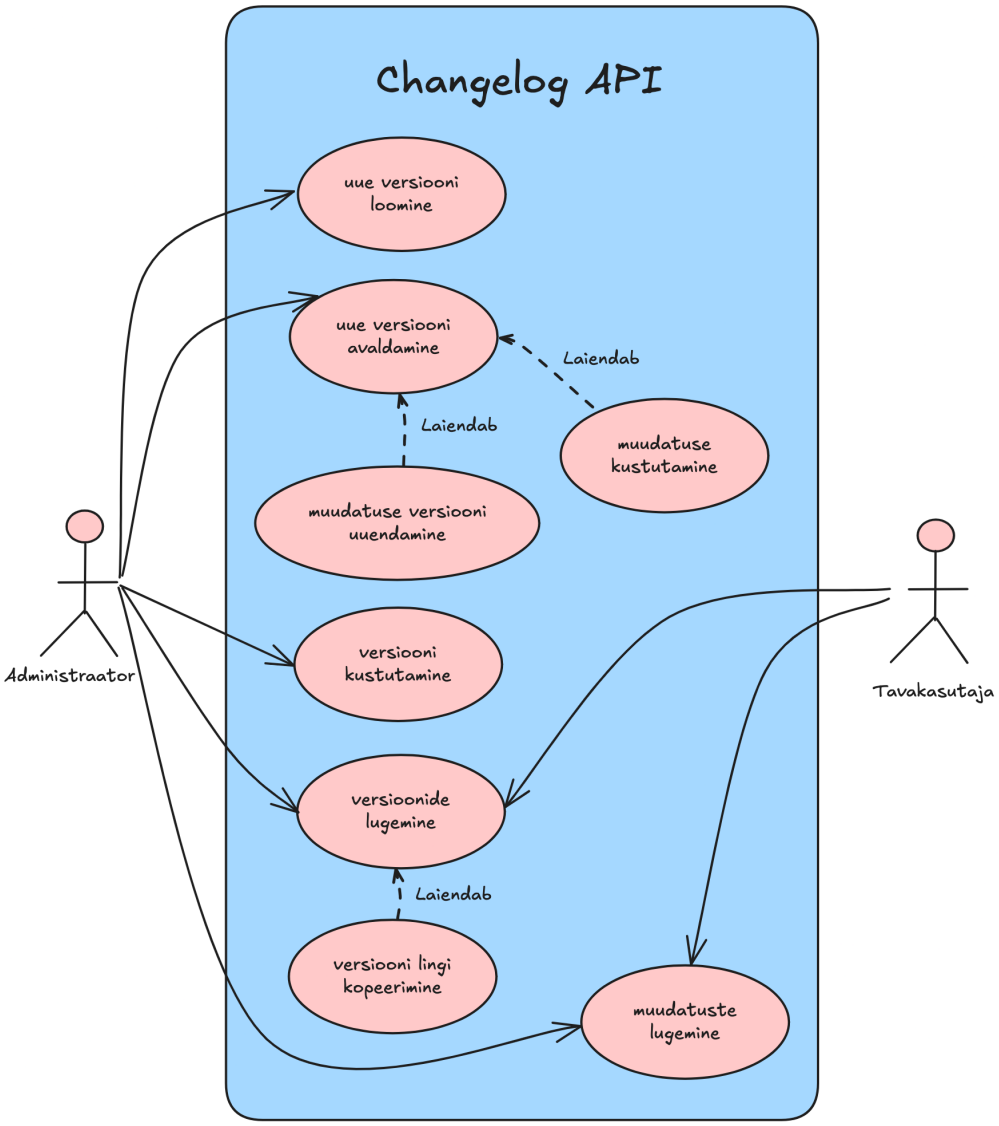
1. Tavakasutaja läheb projekti lehele veebibrauseris.
2. Süsteem kuvab maksimaalselt neli versioonikaarti.
3. Tavakasutaja vajutab soovitud versioonikaardile.
4. Süsteem kuvab valitud versioonile vastavad muudatustekaardid.

Lisatöövood:

1a. Projekti ID ei vasta olemasolevale projektile.
    1. Süsteem vastab veateatega.

2a. Leidub viies versioon.
    1. Tavakasutaja vajutab nupule, mis viib järgmisele lehele.
    2. Süsteem kuvab järgmised neli versioonikaarti.

2b. Tavakasutaja otsustab jagada ühe versiooni muudatusi teiste inimestega.
    1. Tavakasutaja vajutab versioonikaardil linkimise nupule.
    2. Süsteem sisestab versiooni lingi tavakasutaja lõikelauale (ingl *clipboard*).
    3. Süsteem kuvab ajutiselt tavakasutajale lingi kopeerimise kinnitussõnumi.



Joonis 7. Kasutusmalliskeem

2.2 Nõuded

Võttes arvesse muudatuste logimise häid tavasid, sai Changelog API funktsionaalsus defineeritud läbi funktsionaalsete nõuete. Mittefunktsionaalsed nõuded aga annavad piirangud, milles süsteem toimima peab. Järgnevalt esitatud nõuded tagavad, et muudatuste logi on selge ja võimaldab kasutajatel hõlpsasti jälgida tarkvara arengut. Changelog API hakkab kasutama semantilise versioonimise kolmest numbrist (joonis 1) koosnevat versiooni, sest eelistatakse lühemat kirjapilti.

2.2.1 Funktsionaalsed nõuded

Administraator peab saama versioone luua, avaldada, lugeda ja kustutada.

Administraator ei saa kustutada avaldatud versioone.

Administraator saab kustutada avaldamata versioone.

Administraator saab kustutada avaldamata versioonide muudatusi.

Administraator saab liigutada avaldamata versiooni muudatust teise avaldamata versiooni.

Rakendus keelab kasutada ühe projekti raames mitut sama versiooninumbrit.

Rakendus peab salvestama versiooni loomise kuupäeva.

Versioonid peavad olema sorteeritud versiooninumbrite järgi.

Tavakasutaja veebilehel kuvatakse sama tüüpi muudatused lähestikku.

Rakendus peab lubama ainult järgnevaid muudatuse tüüpe:

- lisatud (ingl *added*),
- muudetud (ingl *changed*),
- soovimatuks kuulutatud (ingl *deprecated*),
- eemaldatud (ingl *removed*),
- parandatud (ingl *fixed*),
- turvalisemaks muudetud (ingl *security*).

Tavakasutaja peab saama kopeerida linki kindla versiooni muudatustele.

Rakendus lubab kolmest täisarvust koosnevat versiooninumbrit.

Tavakasutaja peab nägema muudatuste autoreid.

Versioonide lugemine peab olema lehekülgjaotusega (ingl *pagination*).

2.2.2 Mittefunktsionaalsed nõuded

Tavakasutaja HTML-leht peab laadima kahe sekundi jooksul tavavõrgutingimustes.

Üheksa administraatorkasutajat peavad samal ajal saama luua uusi versioone oma projektidesse.

Rakenduse logid peavad säilima vähemalt kolm päeva.

Rakendus peab logima versiooni avalikustamiseks kasutatud võtme ID.

3 Valminud rakendus

Rakenduse lähtekood ja API klient koos käsurearakendusega on avalikult leitavad GitHubis kahel aadressil:

- `https://github.com/e1004/Changelog-API`,
- `https://github.com/e1004/Changelog-API-client`.

Programmeerimiskeeleks valiti eelnevale kogemusele tuginedes Python. Keel pakub laia valikut teeke ning lihtsasti mõistetavat süntaksit. Alternatiivina oleks veebiarenduseks sobinud ka Go, Java või PHP. Serveri HTTP-päringuteks kasutati teeki Flask. Kaaluti teisi populaarseid veebiraamistikke Django ja FastAPI, kuid Django paistis paljude lisaväärtust pakkuvate võimaluste poolest sobivat suuremale projektile kui Changelog API [6]. FastAPI projekt on noorem kui Flask ja Django ning sellest tulenevalt viimased kaks näisid olevat stabiilsemad valikud [6].

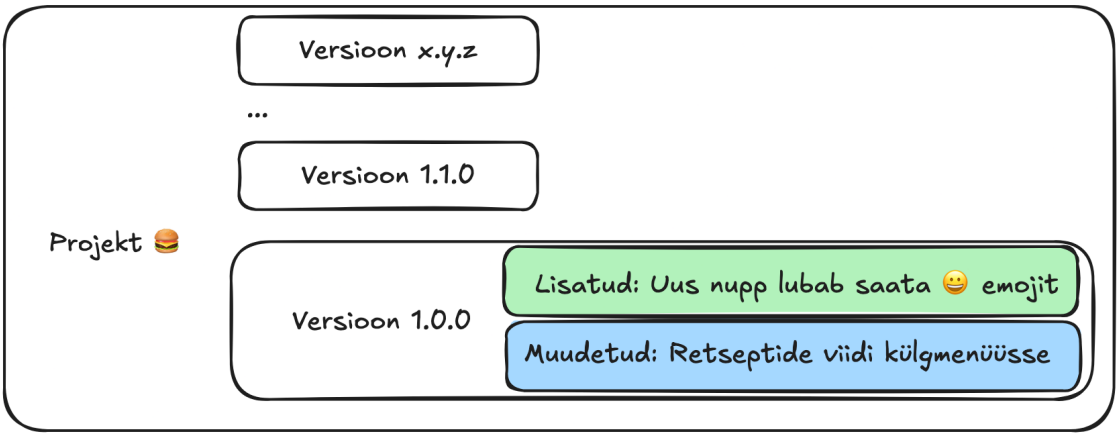
Kuna Changelog API on veebipõhine teenus, siis kasutajad ei pea seda installeerima. Rakenduse kodulehele loodi abistav dokumentatsiooni alamleht (lisa 2). Seal antakse juhised, kuidas seadistada käsurearakendust, et sellega hallata projekte, versioone ja muudatusi.

3.1 Avalik osa

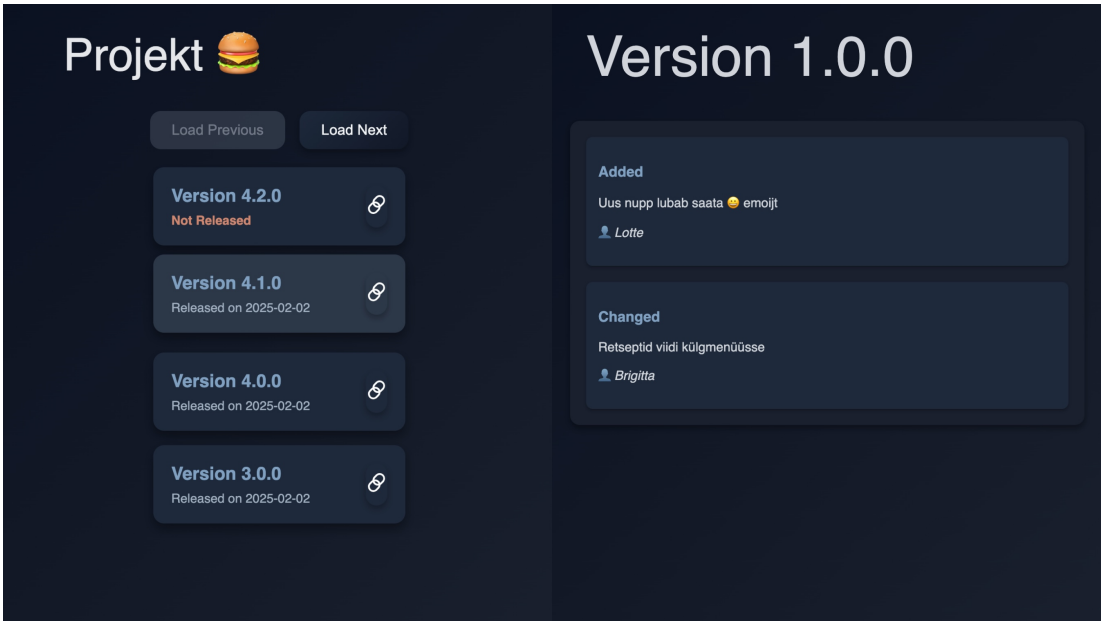
Muudatuste logi avalik osa on HTML-leht, mille idee joonisel 8 võttis arvesse põhiliste andmeklasside omavahelist seost. Muudatused kuuluvad versioonidesse, versioonid omakorda projektidesse. Igal projektil on avaleht versiooni numbrite järgi sorteeritud versioonidega (joonis 9). Kui kõik versioonid visuaalselt ühele lehele ära ei mahu, siis saab tavakasutaja vajutada nuppe, et liikuda versioonide vahel edasi-tagasi. Kasutajaliideses on esile tõstetud avaldamata versioonid ning avaldatud versioonide väljalaske kuupäevad.

Muudatuste vaade joonisel 9 esitab tavakasutajale versiooninumbri koos sinna kuuluvate muudatustega. Iga muudatuse juures on välja toodud muudatuse tüüp, kirjeldus ja autor. Viimase muust tekstist eristamiseks on kasutatud inimese silueti (ingl *silhouette of person*) emotikoni.

Tavakasutaja HTML-lehtede näitamiseks on kaks lõpp-punkti. Flask teegiga kaasa tulev Jinja mallimootor (ingl *templating engine*) lubab muutujaid HTML sisus näidata. Tänu sellele on võimalik kasutajaliidese kood hoida serveri koodiga samas projektis, ilma et oleks vaja hallata kahte eraldi projekti.



Joonis 8. Põhilised seosed



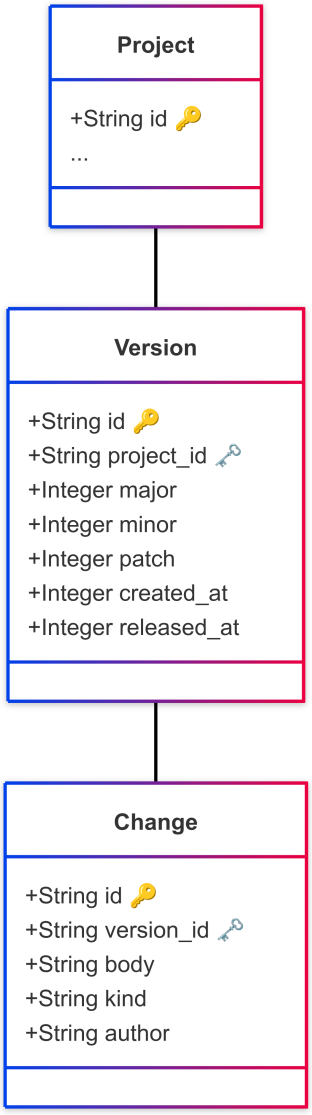
Joonis 9. Kasutajaliides. Vasakul on versioonide vaade, paremal muudatuste vaade

3.2 Andmebaas

Rakenduse andmebaasis on kaks tabelit: versioon ja muudatus (joonis 10). Versiooni tabel omakorda viitab projekti tabelile, mille haldamise eest vastutab projektihalduse teek[4]. Andmebaasiks on SQLite fail. SQLite on rahvusvahelise tiimi poolt hallatav teek, milles on usaldusväärseks töötluseks vajalikud omadused (atomaarsus, konsistentsus, isoleeritus,

[4] `https://pypi.org/project/realerikrani-project`

püsivus) [7]. Kolmel juhul soovitavad SQLite loojad kasutada klient-server-arhitektuuril põhinevat andmebaasi: kui andmed ja veebirakendus asuvad erineval seadmel, kui andmeid kirjutatakse mitme protsessi poolt konkurentselt ning kui andmete maht kasvab terabaidini [7]. Kuna Changelog API plaanitakse tarnida andmebaasifailiga samasse serverisse, ei oodata üle 10 samaaegse andmekirjutusoperatsiooni ja eeldatakse esialgu, et andmete maht jääb alla terabaidi, siis peeti mõistlikuks kasutada SQLite'i.



Joonis 10. Andmebaasi tabelid

Versiooni tabelisse salvestatakse järgnevad väljad:

- projekti ID (võõrvõti),
- versiooni ID (primaarvõti),
- peaversiooninumber (ingl *major*),
- väikeversiooninumber (ingl *minor*),
- parandusversiooninumber (ingl *patch*),
- loomiskuupäev,
- väljalaskekuupäev.

Versiooninumber jagati kolme erineva välja vahel, sest siis sai versioonilugemispäringutes sorteerida täisarvude järgi SQL-päringuid: `ORDER BY major DESC, minor DESC, patch DESC`. Vastasel juhul nende arvude ühe väljana salvestamine oleks sundinud kasutama sõne tüüpi (*TEXT*). Versiooniloomiskuupäev muudeti kohustuslikuks. Samas väljalaskekuupäev on lubatud jätta määramata, sest peab olema võimalik valmistada ette tulevikuversioone.

Muudatuste tabelisse salvestatakse viis välja:

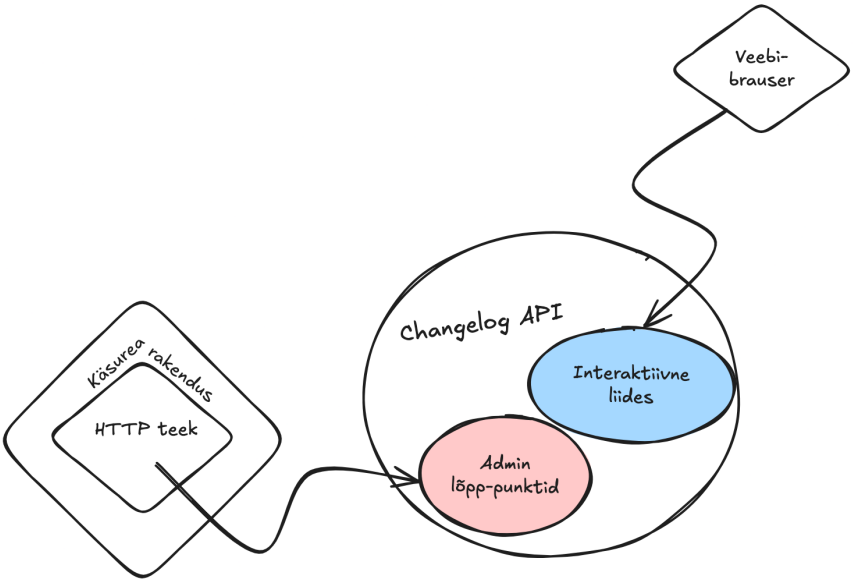
- muudatuse ID (primaarvõti),
- versiooni ID (võõrvõti),
- muudatuse kirjeldus,
- muudatuse tüüp,
- muudatuse autor.

Muudatuste tüübid võivad olla *added*, *changed*, *deprecated*, *removed*, *fixed*, *security*. Nende väärtuste andmebaasitasemel piiramiseks on kasutusel *CHECK* SQL-võtmesõna. Muudatuse kirjelduse tähemärkide piiranguks sai valitud 1000, sest liiga pikk tekst võib teha muudatustelogi raskesti loetavaks.

Koodi struktureerimisel lähtuti kihilisuse põhimõttest. Andmebaasi haldusega tegeleb hoidlakiht, kus on defineeritud SQL-päringud. Kasutaja API-päringute või HTML-lehtede serveerimise eest vastutab kontroller. Kontrollerist tulevaid käske kontrollib üle teenusekiht, mis viskab valideerimisega seotud veateateid ning laseb läbi hoidlakihis ilmnenud probleeme. Kontrolleris hallatakse kõiki etteaimatavaid veajuhtumeid neid HTTP veakoodidesse tõlgendades. Haldamata juhtumid muutuvad HTTP 500 (ingl *Internal Server Error*) koodideks.

3.3 Administreerimine

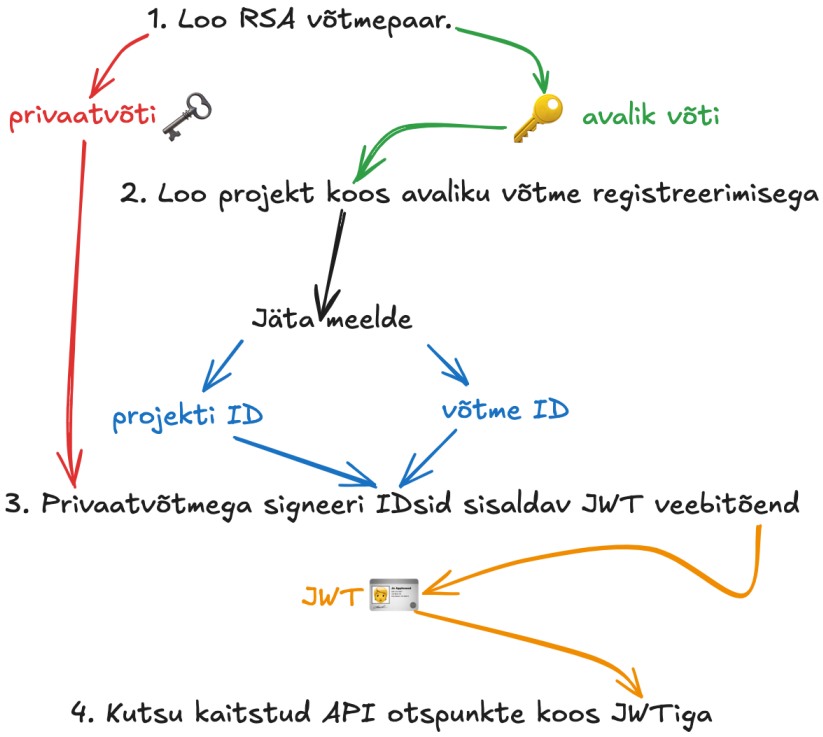
Projekti versioonide vaade ja muudatuste vaade kuuluvad Changelog API interaktiivse liidese alla. Seal näha olevate andmete tekitamiseks on vaja administraatorkasutajal kutsuda andmete haldamise lõpp-punkte (joonis 11). Administraator saab kasutada käsurearakendust[5], Pythonis kirjutatud teeki või tavalisi veebipäringuid. Käsurearakendus ise kasutab seda Pythoni teeki, mis omakorda teeb päringuid Changelog API-le. See on loodud muutmaks kasutajakogemust mugavamaks.



Joonis 11. Kaks Changelog API liidest

Changelog API administraatori lõpp-punktid on kaitstud asümmeetrilise krüptograafiaga (joonis 12). Neid tuleb kutsuda välja koos privaatvõtmega signeeritud JWT (*JSON Web Token*) veebitõendiga. JWTi seest kontrollib rakendus projekti ja võtme identifitseerijaid. Projekti loomisega koos tuleb alati registreerida RSA avalik võti. Hiljem on võimalik soovi korral ka lisavõtmeid registreerida. Võtmepaari genereerimiseks sobib OpenSSL teegi käsk `genpkey`. Changelog APIs on kirjeldatud projektihalduse funktsionaalsus tänu välisele teegile[6].

[5] `https://github.com/e1004/Changelog-API-client`
[6] `https://pypi.org/project/realerikrani-project`



Joonis 12. Võtmete roll

Changelog API administreerimiseks loodi esimeses versioonis kaheksa lõpp-punkti:

- GET `/versions` – versioonide lugemine,
- POST `/versions` – versiooni loomine,
- DELETE `/versions/<version_number>` – versiooni kustutamine,
- PATCH `/versions/<version_number>` – versiooni väljalaskekuupäeva uuendamine,
- GET `/versions/<version_number>/changes` – versiooni muudatuste lugemine,
- POST `/versions/<version_number>/changes` – versiooni muudatuse loomine,
- DELETE `/versions/<version_number>/changes/<change_id>` – versiooni muudatuse kustutamine,
- PATCH `/versions/<version_number>/changes/<change_id>` – versiooni muudatuse liigutamine teise versiooni.

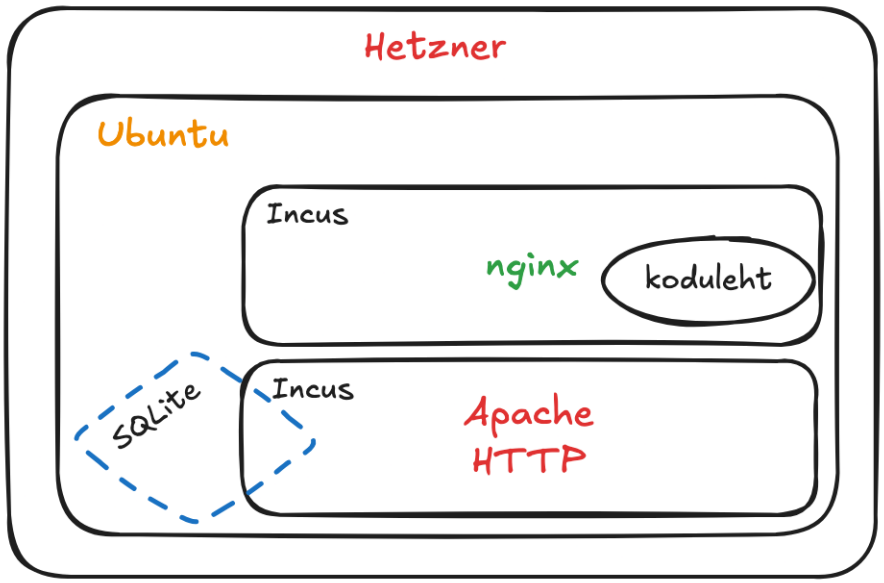
POST- ja PATCH-päringuid tehes peab kasutaja JSON-formaadis andma kaasa päringukeha. Versiooni loomisel eeldatakse kehas versiooninumbrit. Versiooni väljalaskekuupäeva uuendamisel peab seadma selle kuupäeva ISO 8601 formaadis, näiteks 2025-03-01 esimese märtsi esitamiseks. Versiooni muudatuse loomiseks on vaja autori nime, muu-

datuse kirjeldust ja muudatuse tüüpi. Versiooni muudatuse liigutamisel teise versiooni tuleb URLis täpsustada hetkel muudatust sisaldav versiooninumber ja päringukehas uus versiooninumber.

3.4 Süsteemihaldus

Changelog API esimene versioon pandi üles Hetzneri pilveteenuse Ubuntu virtuaalmasinasse (joonis 13). Seda Apache HTTP serveris taustal deemonina jooksvat protsessi kaitsevad Hetzneri ja Ubuntu tulemüürid. Võrguliiklust suunab nginx kas kodulehele või rakendusse. Kui URLis on kirjas `/app`, siis läheb päring APIle. Nginx ja Apache HTTP on mõlemad Incuse Linuxi süsteemikonteinerites, mis on eraldatud virtuaalmasina operatsioonisüsteemist [8].

Rakenduse SQLite andmebaas asub Incuse konteineri ja virtuaalmasina failisüsteemi jagatud kaustas. Iga päev varundatakse andmebaasifaili kindlaks määratud kellaajal Systemd taimeri abil. Systemd taimer on Ubuntus olemasolev plaanur, mis lubab defineerida vajaliku käsu jooksutamist [9].



Joonis 13. Süsteemihaldamise komponendid

Turvalisuse tagamiseks on kasutusel mitmed meetmed. HTML-lehte vaatavat kasutajat kaitstakse päringu vastustega kaasa minevate päistega:

- `X-Frame-Options "DENY"` – kaitseb klõpsurööviv (ingl *clickjacking*) eest

- `X-Content-Type-Options "nosniff"` – kaitseb MIME-nuuskimise rünnaku eest
- `X-XSS-Protection "1; mode=block"` – kaitseb skriptisüstimise (ingl *Cross-Site Scripting*) eest
- `Content-Security-Policy "default-src 'self';"` – blokeerib väljaspool Changelog API lehe pordi, domeeni ja protokolli asukohta oleva sisu laadimise.

Nginx tasemel keelatakse ära kõigi päringute tegemine HTML-lehtedele HTTP päringutega, mis pole GET või POST (koodikatkend 1). Kui nginx suunab päringu rakendusse, siis lubab see seal teha ainult järgnevaid päringuid: GET, POST, DELETE ja PATCH. Changelog APIt jooksutavas Apache HTTP serveris on sisse lülitatud veebitulemüür (ingl *Wed Application Firewall*) ModSecurity. Selle OWASP CRS (*Open Worldwide Application Security Project Core Rule Set*) üldistes rünnakutuvastus meetodite reeglites oli samuti tarvis lubada samad HTTP päringumeetodid [10].

Teenusetõkestusrünnaku (ingl *Denial of Service*) eest kaitsevad rakendust nginx ja Apache ModSecurity. Muudatuste logi avalikule osale saab ühelt IP-aadressilt teha korraga viis samaaegset ühendust ning maksimaalselt viis päringut sekundis. Nginx *burst nodelay* kaudu on lubatud lisaks panna puhvrisse 150 päringut ning neid esimesel võimalusel korraga töödelda, ületamata sagedust viis päringut sekundis uute peale tulevate päringute puhul [11]. Changelog API Pythoni serverile kehtivad ModSecurity piirangud. Päringu tegija blokeeritakse ära 600 sekundiks, kui ta teeb rohkem kui 100 päringut 60 sekundi jooksul kaks korda.

```
limit_conn_zone $binary_remote_addr zone=max-simultaneous-connections:10m;
limit_req_zone $binary_remote_addr zone=allow-burst-for-5-per-second:10m rate=5r/s;

server {

    location / {
        limit_except GET POST{ deny all; }
        limit_conn max-simultaneous-connections 5;
        limit_req zone=allow-burst-for-5-per-second burst=150 nodelay;
    }

    location /app {
        rewrite ^/app/(.*)$ /$1 break;
        limit_except GET POST DELETE PATCH { deny all; }
    }

}
```

Koodikatkend 1. Nginx turvaseaded

3.5 Lõpptulemus ja vastavus nõuetele

Changelog API programmeerimiskeeles Python kirjutatud kood kaeti testidega üle 90 protsendi ulatuses. Kaetavust kontrolliti nii lausete kui harude arvestuses `pytest-cov` teegiga. Koodi loetavuse parandamiseks lisati juurde tüübiannotatsioonid, mille õigsust tagati teegiga `mypy`. Tööriistaga `ruff` sooritati automaatset lähtekoodi vormindamist. Changelog API kliendi ning tavakasutaja HTML-lehtede kasutajaliidese funktsionaalsuseid testiti ainult käsitsi. Dokumentatsiooni kirjutamisega samal ajal prooviti läbi olulised töövood, mida võrreldi funktsionaalsete nõuetega.

Tavakasutaja HTML-lehe laadimiskiirus mittefunktsionaalsete nõuete kohaselt pidi olema maksimaalselt kaks sekundit. Mõõtmine teostati näidisandmetega tööriistaga `https://www.webpagetest.org`. Testi seadetes valiti mõõtmisasukohaks Frankfurt, veebilehitsejaks Chrome ja internetiühenduseks 4G Fast maksimaalse allalaadimiskiirusega 10 Mbps. Sooritati kaheksa katset ekraanisuurusega 1366x768. Lubati kordusvaaterežiim, mis avas lehe, sulges veebilehitseja ning laadis lehe uuesti. Testi tulemustel esimese korra laadimiskiirus oli umbes 0.6 sekundit ja teise korra 0.4 sekundit (joonis 14).

Joonis 14. WebPageTest lehe testi tulemused

Kontrolliti mittefunktsionaalset nõuet, mille kohaselt peavad üheksa administraator-kasutajat saama luua samaaegselt uue versiooni oma projektidesse. Selleks valmistati ette

üheksa projekti. Seejärel loodi üheksa terminalivaadet tööriistaga `tmux`. Igasse vaatesse seadistati keskkonnamuutujate abil eraldi vastava projekti võtmed. Samaaegselt käivitati `tmuxi` vaadete sünkroniseerimisrežiimis uue versiooni loomise käsk. Mittefunktsionaalne nõue osutus täidetuks, sest versioonid loodi edukalt (joonis 15).

Joonis 15. Samal ajal uute versioonide loomine

Veel üks nõue oli seotud logimisega. Iga versiooni avalikustamisel on vaja logida autentimiseks kasutatud privaatvõtmele vastava avaliku võtme ID. Selleks lisati rakendusse vastava API lõpp-punkti logimise käsk ning seadistati serveris logide asukoht. Kontrollimiseks tehti läbi versiooni avalikustamise töövoog ning veenduti, et nõutud info sai talletatud logidesse. Viimane mittefunktsionaalne nõue nägi ette, et rakenduse logid säilitatakse vähemalt kolm päeva. Apache seadistati programmiga `logrotate` alles jätma viimase 14 päeva rakenduse logisid.

4 Kokkuvõte

Bakalaureusetöö eesmärk oli luua veebipõhine muudatuste logimise teenus. Sooviti, et rakenduse kasutajad järgiks kindlaid muudatuste logimise reegleid. Selleks analüüsiti olemasolevaid teadusartikleid, veebipõhiseid ressursse ning võrreldi avatud lähtekoodiga projektide logisid. Tehtud analüüsi põhjal leiti, et muudatuste logimise veebiteenuses on vajalik autori nime mainimine, versioonide sorteerimine, info grupeerimine, versioonide linkimine ja semantilise versioonimise numbriformaat. Tuvastati kaks kasutajatüüpi: administraator ja tavakasutaja, kus esimesel on õigused sisuloomiseks ning teisel selle tarbimiseks.

Praegu saab Changelog API rakendust administreerida otse läbi HTTP veebipäringute. Üks ülesanne oli muuta muudatuste loomine ligipääsetavamaks neile, kellele HTTP veebipäringute saatmine võib keeruline tunduda. Selleks loodi veebiteenuse API klient koos käsurearakendusega. Viimase kasutusmugavust saaks tulevikus parendada: tuua sisse värvikoodid, esitada kasutajasõbralikumaid veateateid ning lisada käskude automaatjätkamine.

Tavaliselt kasutajad vajavad abi uue tarkvara kasutuselevõtuks. Töö käigus loodi rakenduse kodulehele abistav dokumentatsioon, mis annab juhised uute projektide, versioonide ning muudatuste haldamiseks. Tulevikus võiks lisada veebilehele detailsed OpenAPI reeglitele vastavad API lõpp-punkte kirjeldavad Swagger UI vaated läbi HTML `iframe` siltide. Lisaks on käsurearakenduse puhul vaja kodulehe kasutusjuhendis loetleda kõik võimalikud käsud ja parameetrid. Kasulik oleks ka kirjutada õpiku (ingl *tutorial*) stiilis artikleid, mis viiksid algaja muudatuste logija teadmised semantilisest versioonimisest ja muudatuste logimisest uuele tasemele.

Edasiarendusena on võimalik luua administreerimiseks graafilise kasutajaliidesega rakendus. See lubaks luua uusi versioone ja muudatusi mõnede inimeste jaoks lihtsamal viisil. Hetkel pakutav lahendus eeldab tehnilist vilumust. Näiteks saaks kasutajad sisse logida oma kasutajanime ja parooliga ning ühendada nendega olemasolevad RSA-võtmetepõhised projektid.

Viidatud kirjandus

[1] Wu J., He H., Gao K., Xiao W., Li J. ja Zhou M. „A comprehensive analysis of challenges and strategies for software release notes on GitHub“: *Empirical Software Engineering: An International Journal*, 2024, URL: https://research-ebsco-com.ezproxy.utlib.ut.ee/linkprocessor/plink?id=6677cb9e-3b56-30cf-8f7e-ea2ede81b853. (26.11.2024).

[2] Chen Kai, Schach Stephen R., Yu Liguo, Offutt Jeff ja Heller Gillian Z. „Open-Source Change Logs“: *Empirical Software Engineering*, 2004, URL: https://research-ebsco-com.ezproxy.utlib.ut.ee/linkprocessor/plink?id=7bee3ffb-b947-3f02-89d0-d55c53fc27b2. (26.11.2024).

[3] Preston-Werner Tom. *Semantic Versioning 2.0.0*. URL: https://semver.org. (26.11.2024).

[4] Ahlbrandt J., Bott C., Moll P., Naziyok T., Majeed R. W. ja Röhrig R. „Version Changes in Medical Software: Proposing Minimal Requirements for Release Notes and a Version Number Convention — An Operators' Point of View“: *Studies in Health Technology and Informatics*, 2015, URL: https://research-ebsco-com.ezproxy.utlib.ut.ee/linkprocessor/plink?id=3a45ebe4-0cae-39b5-84d5-ce5a1c07f07c. (26.11.2024).

[5] Lacan O. *Keep a Changelog*. URL: https://keepachangelog.com/en/1.1.0. (26.11.2024).

[6] Verbina E. *Which Is the Best Python Web Framework: Django, Flask, or FastAPI?*. URL: https://blog.jetbrains.com/pycharm/2025/02/django-flask-fastapi/. (08.05.2025).

[7] *SQLite*. URL: https://www.sqlite.org. (13.02.2025).

[8] *Incus*. URL: https://linuxcontainers.org/incus. (09.03.2025).

[9] Both D. *Use systemd timers instead of cronjobs*. URL: https://opensource.com/article/20/7/systemd-timers. (09.03.2025).

[10] *OWASP CRS*. URL: https://owasp.org/www-project-modsecurity-core-rule-set. (12.03.2025).

[11] Rawdat A. *Rate Limiting with NGINX*. URL: https://blog.nginx.org/blog/rate-limiting-nginx. (18.03.2025).

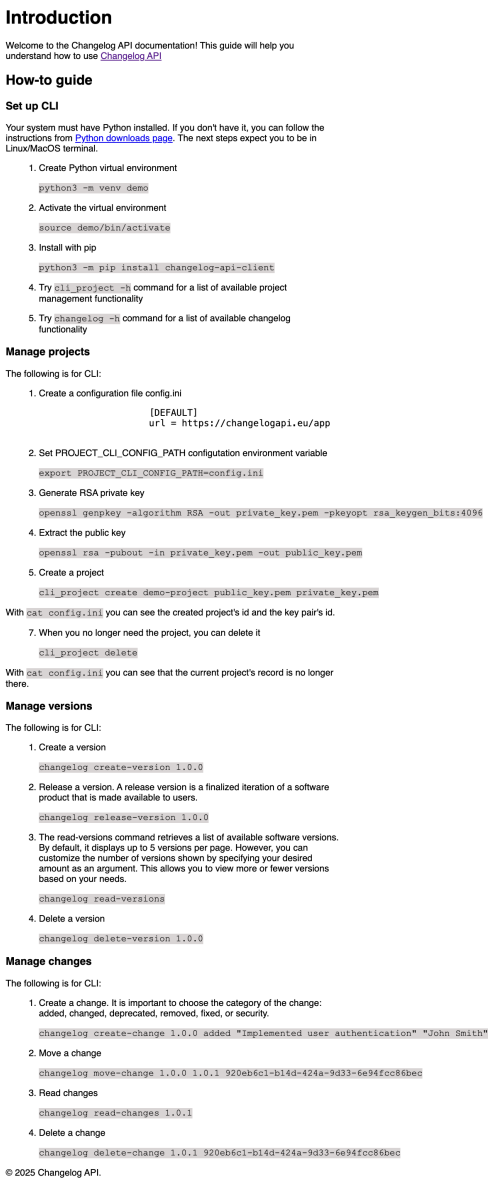
Lisad

I. Avatud lähtekoodiga rakendused

Tabel 3. Avatud lähtekoodiga rakenduste muudatuste logi asukohad

Nimi	**Link**
React	https://github.com/facebook/react/blob/main/CHANGELOG.md
Vue.js	https://github.com/vuejs/vue/blob/main/CHANGELOG.md
Angular	https://github.com/angular/angular/blob/main/CHANGELOG.md
Flask	https://flask.palletsprojects.com/en/latest/changes/
Node.js	https://github.com/nodejs/node/blob/main/CHANGELOG.md
MySQL	https://dev.mysql.com/doc/relnotes/mysql/8.0/en/
Nginx	https://nginx.org/en/CHANGES
Next.js	https://github.com/vercel/next.js/releases
MongoDB	https://www.mongodb.com/docs/manual/release-notes
PostgreSQL	https://www.postgresql.org/docs/release/
Homebrew	https://github.com/Homebrew/brew/blob/master/CHANGELOG.md
Ansible	https://github.com/ansible-community/ansible-build-data/blob/main/11/CHANGELOG-v11.md
Apache	https://httpd.apache.org/docs/2.4/new_features_2_4.html
Docker	https://docs.docker.com/engine/release-notes/
Prometheus	https://github.com/prometheus/prometheus/blob/main/CHANGELOG.md
Kubernetes	https://github.com/kubernetes/kubernetes/tree/master/CHANGELOG

II. Kodulehe dokumentatsioon



Introduction

Welcome to the Changelog API documentation! This guide will help you understand how to use Changelog API

How-to guide

Set up CLI

Your system must have Python installed. If you don't have it, you can follow the instructions from Python downloads page. The next steps expect you to be in Linux/MacOS terminal.

1. Create Python virtual environment

   `python3 -m venv demo`

2. Activate the virtual environment

   `source demo/bin/activate`

3. Install with pip

   `python3 -m pip install changelog-api-client`

4. Try `cli_project -h` command for a list of available project management functionality

5. Try `changelog -h` command for a list of available changelog functionality

Manage projects

The following is for CLI:

1. Create a configuration file config.ini

   ```
   [DEFAULT]
   url = https://changelogapi.eu/app
   ```

2. Set PROJECT_CLI_CONFIG_PATH configutation environment variable

   `export PROJECT_CLI_CONFIG_PATH=config.ini`

3. Generate RSA private key

   `openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:4096`

4. Extract the public key

   `openssl rsa -pubout -in private_key.pem -out public_key.pem`

5. Create a project

   `cli_project create demo-project public_key.pem private_key.pem`

With `cat config.ini` you can see the created project's id and the key pair's id.

7. When you no longer need the project, you can delete it

   `cli_project delete`

With `cat config.ini` you can see that the current project's record is no longer there.

Manage versions

The following is for CLI:

1. Create a version

   `changelog create-version 1.0.0`

2. Release a version. A release version is a finalized iteration of a software product that is made available to users.

   `changelog release-version 1.0.0`

3. The read-versions command retrieves a list of available software versions. By default, it displays up to 5 versions per page. However, you can customize the number of versions shown by specifying your desired amount as an argument. This allows you to view more or fewer versions based on your needs.

   `changelog read-versions`

4. Delete a version

   `changelog delete-version 1.0.0`

Manage changes

The following is for CLI:

1. Create a change. It is important to choose the category of the change: added, changed, deprecated, removed, fixed, or security.

   `changelog create-change 1.0.0 added "Implemented user authentication" "John Smith"`

2. Move a change

   `changelog move-change 1.0.0 1.0.1 920eb6c1-b14d-424a-9d33-6e94fcc86bec`

3. Read changes

   `changelog read-changes 1.0.1`

4. Delete a change

   `changelog delete-change 1.0.1 920eb6c1-b14d-424a-9d33-6e94fcc86bec`

© 2025 Changelog API.

Joonis 16. Ekraanitõmmis kasutusjuhendist

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Elisabeth Räni**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Changelog API – veebipõhine muudatuste logimise teenus**, mille juhendajad on Erik Räni ja Ljubov Jaanuska, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Elisabeth Räni
14.05.2025