LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

MIKROSERVISU IZSTRĀDE IZKLAIDES PLATFORMAI AIZMUGURĒJĀ SISTĒMĀ

KVALIFIKĀCIJAS DARBS

Autors: Marta Bulāne
Stud. apl. Nr. mb17042
Darba vadītājs: M. dat. Oskars Ozols

RĪGA 2019

ANOTĀCIJA

Kvalifikācijas darbā tika veikta izklaides platformas “Shortcut.lv” mikroservisu izstrāde aizmugurējā sistēmā. Tas sastāv no programmatūras prasību speficikācijas, programmatūras projektējuma un testēšanas dokumentācijas.

Darba mērķis ir izveidot vairāku mikroservisu funkcionalitāti, kas ir jaunās aizmugurējās sistēmas sastāvdaļa.

Mikroservisu izveidē tika izmantota PHP programmēšanas valoda, Lumen ietvars, Apache Cassandra datu bāzu tehnoloģija. Testiem tika izmantots Codeception automatizētās testēšanas ietvars.

Atslēgvārdi: mikroserviss, PHP, Cassandra, automatizētie testi

ABSTRACT

BACK END DEVELOPMENT OF MICROSERVICES FOR THE ENTERTAINMENT PLATFORM

The qualification work consists of entertainment platform's "Shortcut.lv" back end development of microservices. It contains software requirements, software design and testing documentation.

The goal of this work is to develop functionality of several microservices, which are part of the new back end.

PHP programming language, Lumen framework and Apache Cassandra database technology were used in the development of microservices. Codeception automated testing framework was used for tests.

Keywords: microservice, PHP, Cassandra, automated tests

SATURS

IEVADS

“Shortcut.lv” ir uzņēmuma SIA “Tet” izklaides platforma, kur var noskatīties dažādus seriālus, filmas, raidījumus, kā arī skatīties kanālu tiešraides un arhīva ierakstus. Šobrīd platformas aizmugurējā sistēma ir izstrādāta monolīta arhitektūrā. Platformā katru dienu tiek pievienotas jaunas filmas, seriāli u.c., kā arī palielinās lietotāju skaits. Pašreizējā sistēmas arhitektūra var neizturēt lietotāju slodzi un nav pieļaujams, ka platforma nestrādā, ja kāda funkcija ir avarējusi, tāpēc tika pieņemts lēmums pāriet uz modernāku sistēmas arhitektūru - mikroservisu arhitektūru.

Mikroservisi tiek darbināti neatkarīgi, tāpēc gadījumā, ja kāds no tiem ir avarējis, tas neapstādina pārējo sistēmas darbību.

Šī darba ietvaros ir izstrādāta četru mikroservisu funkcionalitāte, kas ir sastāvdaļa no jaunās aizmugurējās sistēmas.

DARBĀ IZMANTOTIE APZĪMĒJUMI

API (Application programming interface) – Apzīmē lietojumprogrammu saskarnes. Programmatūras veidošanas bloki, kas spēj sazināties savā starpā.

Cassandra – Apache Cassandra ir atvērtā koda kolonnu orientēta datu bāze. [5]

Cron – Programmatūras darbu plānotāju utilītprogramma, kas ir balstīta uz konkrētu laika izpildi.

Docker – Atvērtā koda platforma lietotņu automātiskai darbināšanai konteineros.

Epg – Arhīva ieraksts

Git – Atvērtā koda versiju kontroles sistēma.

GitLab – Programmatūras koda versiju kontroles rīks,

HTTP (Hypertext Transfer Protocol) – Lietojumslāņu protokols, kas paredzēts datu apmaiņai starp tīmekļa serveriem un pārlūkprogrammām.

MDS (Meta datu sistēma) – Iekšējā "Tet" sistēmā, kurā tiek glabāti dati.

Mikroservisu arhitektūra – Programmatūras izstrādes arhitektūra, kas strukturē lietotni kā servisu kolekciju.

MySQL – Atvērtā koda relāciju datu bāzes pārvaldības sistēma.

NoSQL – Atvērtā koda izplatīta datu bāze, kas neizmanto SQL saskarni.

PHP (Hypertext Preprocessor) – Atklātā pirmkoda skriptu valoda.

Scrum - Procesu ietvars, kas tiek lietots sarežģītu produktu izstrādē, uzsvaru liekot uz komandas darbu, atbildību un iteratīvu progresu virzienā uz noteiktu mērķi.

TLS (Transport Layer Security) – Šifrēšanas protokols, kuru izmanto, lai šifrētu datortīklos pārsūtīto informāciju.

Video – filma, seriāla epizode vai raidījums, kas ir ievietots "Shortcut.lv" platformā.

1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

1.1. Ievads

1.1.1. Nolūks

Programmatūras prasību specifikācijā (PPS) tiek apkopotas funkcionālās un nefunkcionālās prasības. Šī dokumenta nolūks ir aprakstīt detalizēti izklaides platformas "Shortcut.lv" mikroservisu programmatūras prasības.

1.1.2. Darbības sfēra

Izveidotie servisi ir daļa no jaunās aizmugurējās sistēmas, kas tiek veidota mikroservisu arhitektūrā. Tuvākajā nākotnē ir paredzēts esošo arhitektūru, kas ir monolīts, nomainīt pret modernu un dinamiskāku - mikroservisu arhitektūru. Autores izstrādātie mikroservisi: *LikeDislike, View, SyncEpg* un *SyncChannels.*

- *LikeDislike* mikroserviss atbild par vērtējumu funkcionalitāti.
- *View* mikroserviss atbild par skatījumu reģistrēšanu.
- *SyncEpg* mikroserviss atbild par arhīva ierakstu importu no meta datu sistēmas.
- *SyncChannels* mikroserviss atbild par kanālu saraksta importu no meta datu sistēmas.

1.1.3. Saistība ar citiem dokumentiem

Dokuments ir izstrādāts vadoties pēc LVS 68:1996 "Programmatūras prasību specifikācijas ceļvedis" standarta prasībām.

1.1.4. Pārskats

Programmatūras prasību specifikācija sastāv no 4 nodaļām:

- Ievads – ievadinformācija par dokumenta nolūku, darbības sfēru, saistību ar citiem dokumentiem un dokumentu pārskatu.
- Vispārējs apraksts – informācija par mikroservisiem, produkta perspektīvām, funkcijām, sistēmas lietotājiem un vispārējiem ierobežojumiem.
- Funkcionālās prasības – informācija par mikroservisu funkcionālajām prasībām.
- Nefunkcionālās prasības – informācija par mikroservisu nefunkcionālajām prasībām.

1.2. Vispārējs apraksts

1.2.1. Produkta perspektīva

Izveidotie mikroservisi ir “Shortcut.lv” aizmugurējās sistēmas sastāvdaļa.

1.2.2. Produkta funkcijas

Izstrādāto mikroservisu funkcijas:

- video vērtējuma ievietošana datubāzē, vērtējuma ieraksta dzēšana no datubāzes, atgriešana;
- reģistrēt arhīva ierakstu, video skatījumus;
- importēt arhīva ierakstus no metadatu sistēmas;
- importēt kanālu sarakstu no metadatu sistēmas.

1.2.3. Lietotāju raksturiezīmes

Izveidotajiem mikroservisiem nav gala lietotāja – cilvēka. Servisus izmantos publiskā lietotne – priekšgalsistēma.

1.3. Funkcionālās prasības

Šajā nodaļā tiek aprakstītas mikroservisu funkcionālās prasības, tās ir sadalītas četros mikroservisos – *LikeDislike, View, SyncEpg* un *SyncChannels*. Katrā funkcijā tiek pieņemts, ka vispirms apstrādes laikā tiek validēti visi dati - tiek pārbaudīts, vai ir visi nepieciešamie dati un vai ir pareizais datu tips. Gadījumā, ja trūkst kādi dati vai tiem ir nepareizs formāts, tiek izvadīti kļūdu paziņojumi, kā ‘MISSING PARAMS’ vai ‘WRONG PARAMS’.

Izveidotās funkcijas tiek izsauktas, izmantojot HTTP protokolu, tāpēc funkciju izvade sastāv no funkciju izpildes statusa koda:

- 200 – OK – veiksmīga operācijas izpilde;
- 422 – Unprocessable Entity – saņemtie dati nav korekti;
- 500 – Internal Server Error – iekšējā servera kļūda, operācija netika izpildīta.

Lietotājiem ir piekļuves pilnvaras, tas nozīmē, ka lietotājs ir autorizējies un var izmantot platformu.

1.3.1. LikeDislike

LikeDislike mikroserviss nodrošina iespēju novērtēt video. Vērtējumi ir ‘patīk’ un ‘nepatīk’. Lietotājs var nospiest ‘patīk’ vai ‘nepatīk’, novērtējot video, ja vērtējums jau ir

ievietots, tad to var mainīt, nospiežot pretējo vērtējumu, vai arī vērtējumu noņemt, nospiežot vēlreiz uz jau attiecīgā ievietotā vērtējuma.

1.3.1.1. Vērtējuma ievietošana

1.3.1.1. tabula
Mikroservisa funkcionālā prasība – vērtējuma ievietošana

Funkcija	Vērtējuma ievietošana
Identifikators	1.3.1.1.
Apraksts	Funkcija nodrošina video vērtējuma ievietošanu.
Ievade	POST pieprasījums ar vaicājuma datiem.
Apstrāde	Pēc pieprasījuma tiek pārbaudīts, vai vērtējums jau nav ievietots un datubāzē "user_like_dislike" jau nepastāv lietotāja ieraksts: • ja tas pastāv, tad tiek izpildīta funkcija **1.3.1.2. Vērtējuma dzēšana**. Pēc tam attiecīgais vērtējums un lietotājs ieraksts par ievietoto vērtējumu tiek pievienots datubāzēs.
Izvade	Iespējamie HTTP statusa kodi: • 200, ja operācija veiksmīgi izpildīta; • 422, ja validācijas kļūda; • 500, ja operācija neveiksmīga;

1.3.1.2. Vērtējuma dzēšana

1.3.1.2. tabula
Mikroservisa funkcionālā prasība – vērtējuma dzēšana

Funkcija	Vērtējuma dzēšana
Identifikators	1.3.1.2.
Apraksts	Funkcija nodrošina video vērtējuma dzēšanu.
Ievade	POST pieprasījums ar vaicājuma datiem.
Apstrāde	Pēc pieprasījuma tiek pārbaudīts, vai vērtējums pastāv un datubāzē "user_like_dislike" ir lietotāja ieraksts. Pēc tam attiecīgais vērtējums tiek noņemts un tiek izpildīta funkcija **1.3.1.3. Lietotāja ieraksta dzēšana**.
Izvade	Iespējamie HTTP statusa kodi: • 200, ja operācija veiksmīgi izpildīta; • 422, ja validācijas kļūda;

	• 500, ja operācija neveiksmīga;

1.3.1.3. Lietotāja ieraksta dzēšana

1.3.1.3. tabula
Mikroservisa funkcionālā prasība – lietotāja ieraksta dzēšana

Funkcija	Lietotāja ieraksta dzēšana
Identifikators	1.3.1.3.
Apraksts	Funkcija nodrošina lietotāja ieraksta dzēšanu no datubāzes.
Ievade	POST pieprasījums ar vaicājuma datiem.
Apstrāde	Pēc pieprasījuma tiek atrasts atbilstošais lietotāja ieraksts datubāzē "user_like_dislike" un tiek dzēsts.
Izvade	Iespējamie HTTP statusa kodi: • 200, ja operācija veiksmīgi izpildīta; • 422, ja validācijas kļūda; • 500, ja operācija neveiksmīga;

1.3.1.4. Lietotāja vērtējumu atgriešana

1.3.1.4. tabula
Mikroservisa funkcionālā prasība – lietotāja vērtējumu atgriešana

Funkcija	Lietotāja vērtējumu atgriešana
Identifikators	1.3.1.4.
Apraksts	Lietotājam tiek parādīti konkrētā video vērtējumi.
Ievade	GET pieprasījums ar vaicājuma datiem.
Apstrāde	Datubāzēs "user_like_dislike" un "vod_like_dislike" atrod attiecīgos ierakstus un izvada attiecīgā video vērtējumus.
Izvade	Iespējamie HTTP statusa kodi: • 200, ja operācija veiksmīgi izpildīta; • 500, ja operācija neveiksmīga;

1.3.2. View

View mikroserviss nodrošina iespēju reģistrēt un uzskaitīt gan video, gan epg skatījumus. Skatījumu tiek uzskaitīti, ja lietotājs atver video vai epg.

1.3.2.1. Lietotāja arhīva ieraksta skatījuma reģistrēšana

1.3.2.1. tabula
Mikroservisa funkcionālā prasība – lietotāja arhīva ieraksta skatījuma reģistrēšana

Funkcija	Lietotāja arhīva ieraksta skatījuma reģistrēšana
Identifikators	1.3.2.1.
Apraksts	Funkcija nodrošina arhīva ieraksta skatījuma reģistrēšanu.
Ievade	POST pieprasījums ar vaicājuma datiem.
Apstrāde	Pēc pieprasījuma informācija par lietotāju un attiecīgo arhīva ierakstu tiek ievadīta datubāzē “view_user_epg”.
Izvade	Iespējamie HTTP statusa kodi: • 200, ja operācija veiksmīgi izpildīta; • 500, ja operācija neveiksmīga;

1.3.2.2. Lietotāja video skatījuma reģistrēšana

1.3.2.2. tabula
Mikroservisa funkcionālā prasība – lietotāja video skatījuma reģistrēšana

Funkcija	Lietotāja video skatījuma reģistrēšana
Identifikators	1.3.2.2.
Apraksts	Funkcija nodrošina video skatījuma reģistrēšanu.
Ievade	POST pieprasījums ar vaicājuma datiem.
Apstrāde	Pēc pieprasījuma informācija par lietotāju un attiecīgo video tiek ievadīta datu bāzē “view_user_vod”.
Izvade	Iespējamie HTTP statusa kodi: • 200, ja operācija veiksmīgi izpildīta; • 500, ja operācija neveiksmīga;

1.3.3. Epg

Epg mikroserviss nodrošina konkrētu arhīva ierakstu importēšanu no ārējās meta datu sistēmas. Dati tiek importēti ik pēc noteikta laika, importa skriptu iedarbināšanai izmanto atsevišķi izveidotus CRON uzdevumi (*angl*. cron jobs).

1.3.3.1. Arhīva ierakstu imports no MDS

1.3.3.1. tabula
Mikroservisa funkcionālā prasība – arhīva ierakstu imports no MDS

Funkcija	Arhīva ierakstu imports no MDS
Identifikators	1.3.3.1.
Apraksts	Funkcija nodrošina arhīva ierakstu importu no metadatu sistēmas.
Ievade	Datu bāzes tabulas nosaukums.
Apstrāde	Pēc pieprasījuma sākumā tiek iegūta meta datu sistēmas piekļuves pilnvara, tad tiek izpildīta sinhronizācijas funkcija, kas nodrošina datu ieguvi no meta datu sistēmas, tad atbilstošie dati tiek sakārtoti un ievietoti datu bāzē “epg”.
Izvade	Ja operācija noritējusi neveiksmīgi, tad tiek izvadīts paziņojums: “No data received for epg”. Ja operācija noritējusi sekmīgi, tad tiek izvadīts paziņojums: “Data received for epg”.

1.3.4. Channels

Channels mikroserviss nodrošina kanālu saraksta importēšanu no ārējās meta datu sistēmas. Dati tiek importēti ik pēc noteikta laika, importa skriptu iedarbināšanai izmanto atsevišķi izveidotus CRON uzdevumus (*angl*. cron jobs).

1.3.4.1. Kanālu saraksta imports no MDS

1.3.4.1. tabula
Mikroservisa funkcionālā prasība – kanālu saraksta imports no MDS

Funkcija	Kanālu saraksta imports no MDS
Identifikators	1.3.4.1.
Apraksts	Funkcija nodrošina kanālu saraksta importu no metadatu sistēmas.
Ievade	Datu bāzes tabulas nosaukums.
Apstrāde	Pēc pieprasījuma sākumā tiek iegūta meta datu sistēmas piekļuves pilnvara, tad tiek izpildīta sinhronizācijas funkcija, kas nodrošina datu ieguvi no meta

	datu sistēmas, tad atbilstošie dati tiek sakārtoti un ievietoti datu bāzē "channels".
Izvade	Ja operācija noritējusi neveiksmīgi, tad tiek izvadīts paziņojums: "No data received for channels". Ja operācija noritējusi sekmīgi, tad tiek izvadīts paziņojums: "Data received for channels".

1.4. Nefunkcionālās prasības

1.4.1. Drošība

Visu pieprasījumu veikšanai un datu saņemšanai serverim, jāizmanto droši protokoli, *REST API* izmanto HTTP un atbalsta TLS, tāpēc tīmekļa adresei jāsākas ar *HTTPS*. Lietotāju piekļuves pilnvarām jābūt šifrētām. Visiem parametriem un datiem jātiek validētiem.

1.4.2. Uzturamība

Mikroservisu arhitektūra ir daudz dinamiskāka nekā monolīta arhitektūra. Platformai ar katru dienu pievienojas jauni klienti, tāpēc tai viegli jāmērogojas. Tāpēc arī tika nolemts mainīt datu bāžu pārvaldību no MySQL uz NoSQL datu bāzi, kā Cassandra, kas nodrošina iespēju paplašināt klasteri, kur tiek glabāti dati, kā arī palielināt ātrdarbību. Mikroservisu arhitektūra ir labāk uzturama, jo mikroservisa avārijas gadījumā, tas neietekmē visas sistēmas darbību, kā arī jaunās izmaiņas var ieviest ātrāk.

1.4.3. Pieejamība

Mikroservisi ir pieejami sistēmas lietotājiem tīmekļa vidē, ir nepieciešams interneta pieslēgums. Lietotājs tiešā veidā neizmanto mikroservisus, jo tie ir izstrādāti servera pusē, piemēram, lietotājs nospiež 'patīk' pogu, un pateicoties mikroservisam, vērtējums tiek ierakstīts un uzglabāts datubāzē, lai to pēc tam varētu gan atgriezt, gan nomainīt vai izdzēst.

2. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

2.1. Ievads

2.1.1. Nolūks

Šis dokuments ir programmatūras projektējuma apraksts projektam "Mikroservisu izstrāde izklaides platformai aizmugurējā sistēmā".

2.1.2. Darbības sfēra

Izveidotie servisi ir daļa no jaunās aizmugurējās sistēmas, kas tiek veidota mikroservisu arhitektūrā.

Mikroservisi tiek veidoti izmantojot PHP programmēšanas valodu, Lumen ietvaru, dati tiek glabāti Cassandra datu bāzē.

2.1.3. Saistība ar citiem dokumentiem

Dokuments ir izstrādāts vadoties pēc LVS 68:1996 "Programmatūras prasību specifikācijas ceļvedis" standarta prasībām.

2.1.4. Pārskats

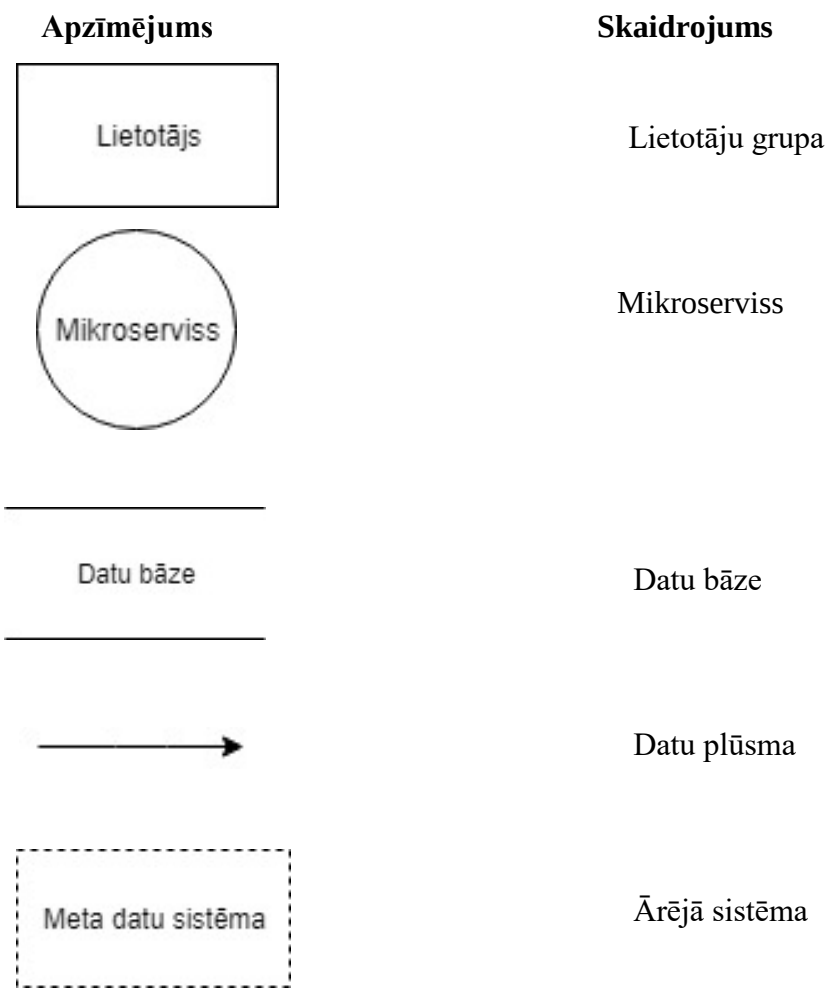
Programmatūras projektējuma apraksts sastāv no 3 daļām:

- Ievads – informācija par dokumenta nolūku, darbības sfēru un saistību ar citiem dokumentiem;
- Datu plūsmu diagrammas – sniedz grafisku attēlojumu par to, kā mikroservisi darbojas;
- Datu bāzes projektējums – informācija par izmantotajām datu bāzes tabulām;

2.2. Datu plūsmas diagrammas

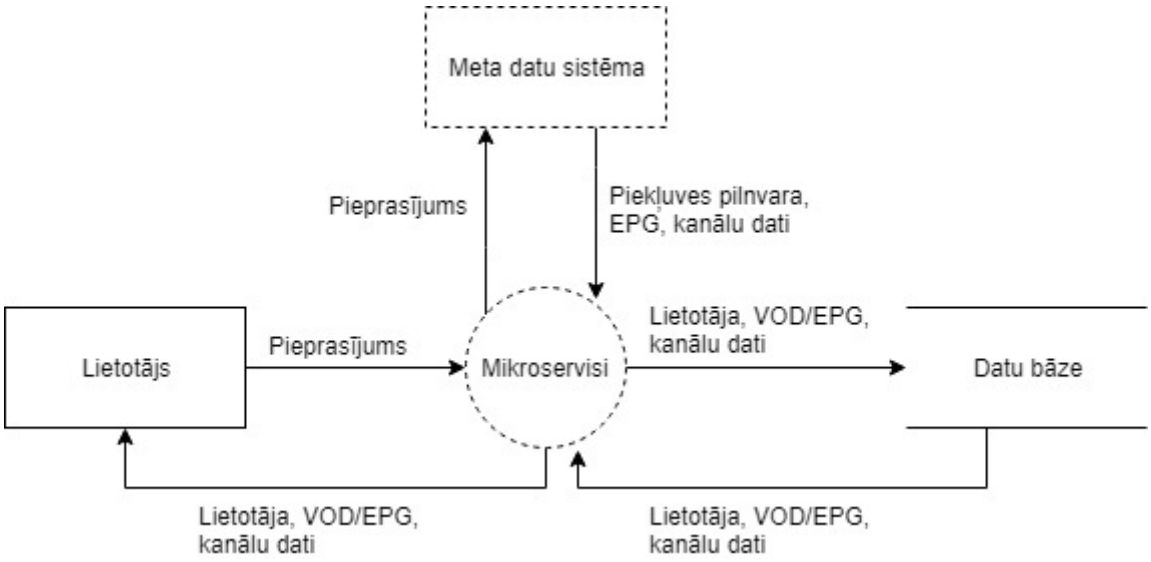
Datu plūsmas diagrammās tiek attēlotas tikai tās daļas, kuras attiecas uz autores izveidotajiem mikroservisiem, netiek attēloti citu izstrādātāju izveidotie mikroservisi.

Datu plūsmu diagrammās izmantotie apzīmējumi:



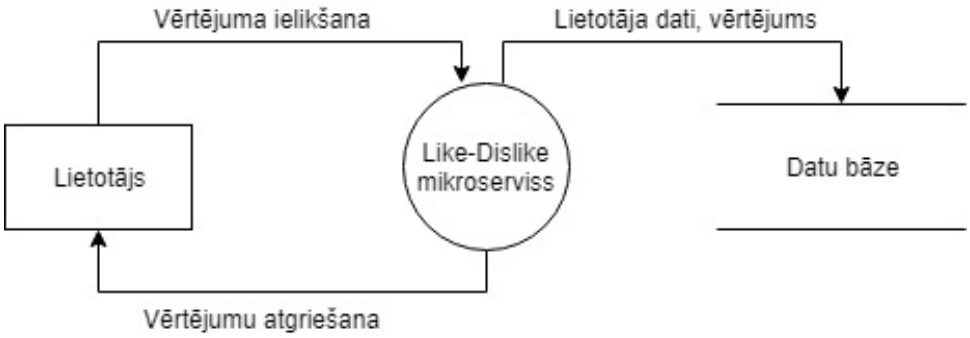
2.2. att. **Datu plūsmas diagrammu apzīmējumi**

2.2.1. 0. līmenis



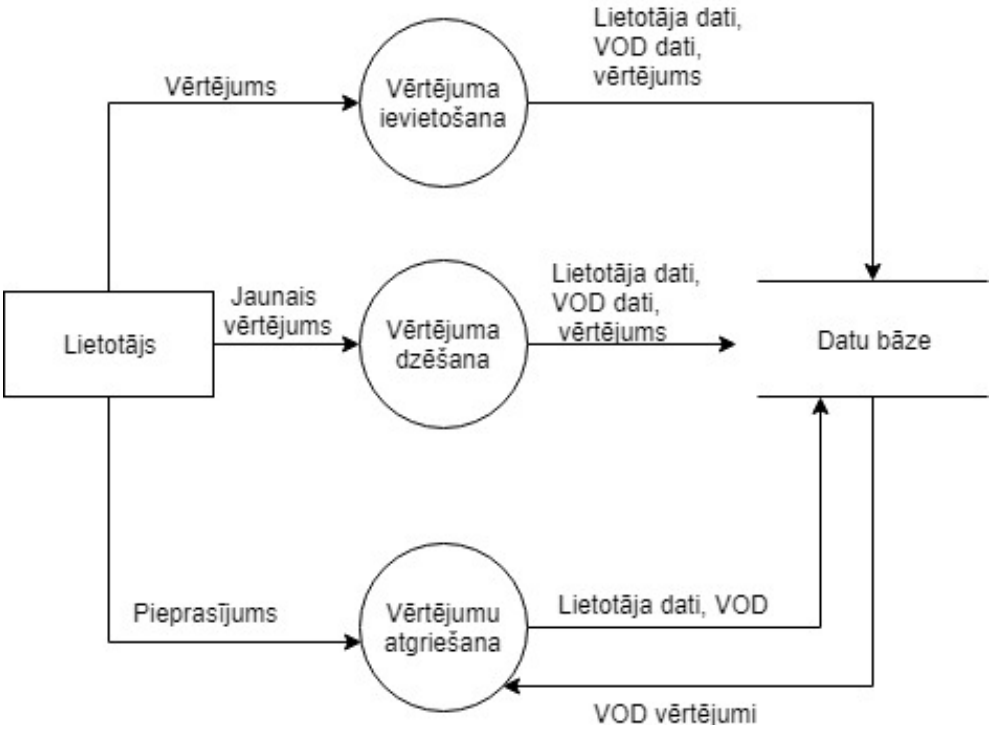
2.2.1. att. **Datu plūsmas diagramma 0. līmenis**

2.2.2. 1. līmenis – LikeDislike



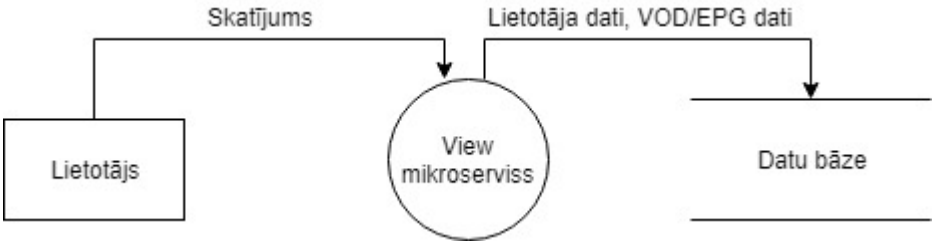
2.2.2. att. **Datu plūsmas diagramma 1. līmenis – LikeDislike**

2.2.3. 2. līmenis – LikeDislike



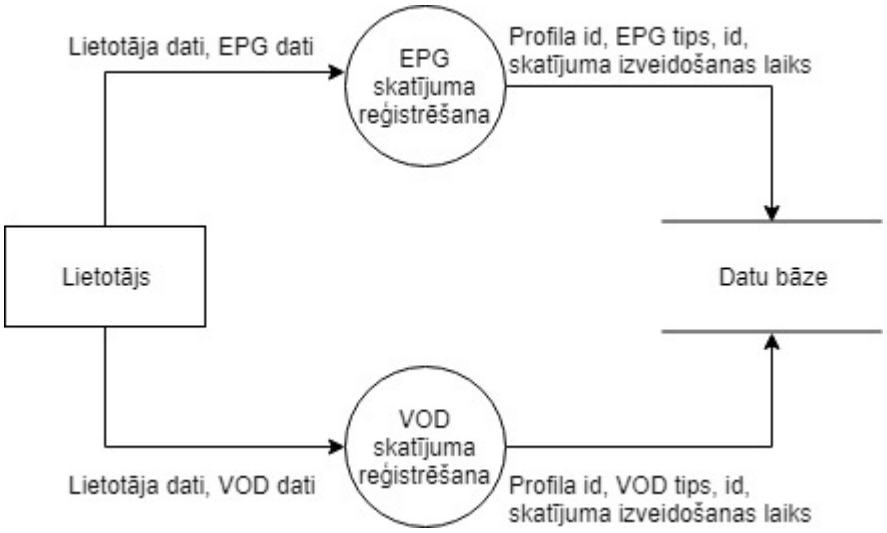
2.2.3. att. **Datu plūsmas diagramma 2. līmenis – LikeDislike**

2.2.4. 1. līmenis – View



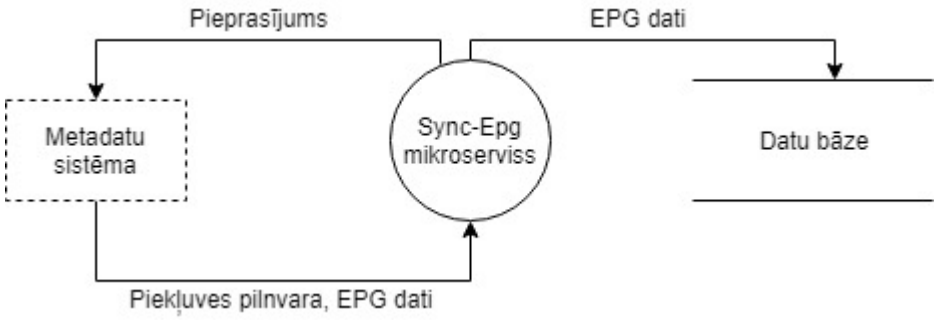
2.2.4. att. **Datu plūsmas diagramma 1. līmenis – View**

2.2.5. 2. līmenis – View



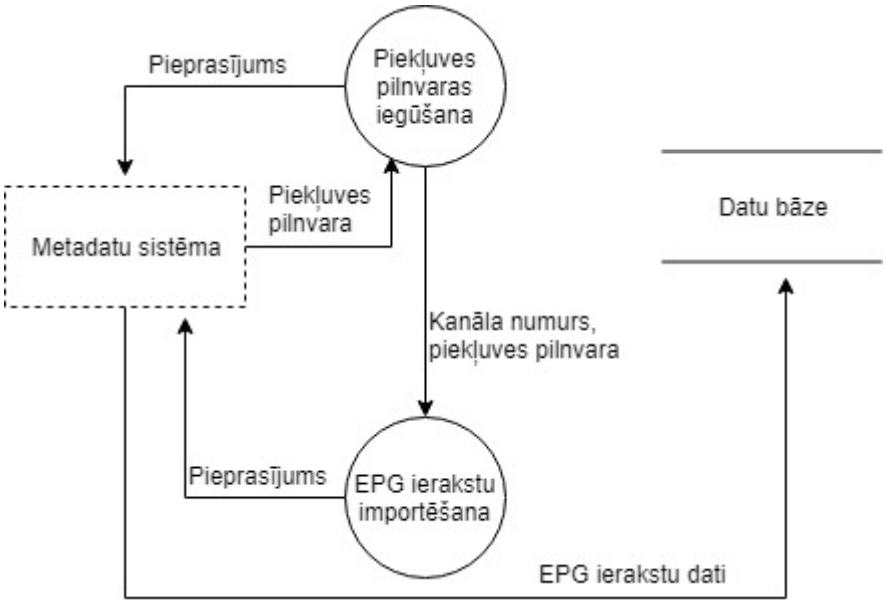
2.2.5. att. **Datu plūsmas diagramma 2. līmenis – View**

2.2.6. 1. līmenis – SyncEpg



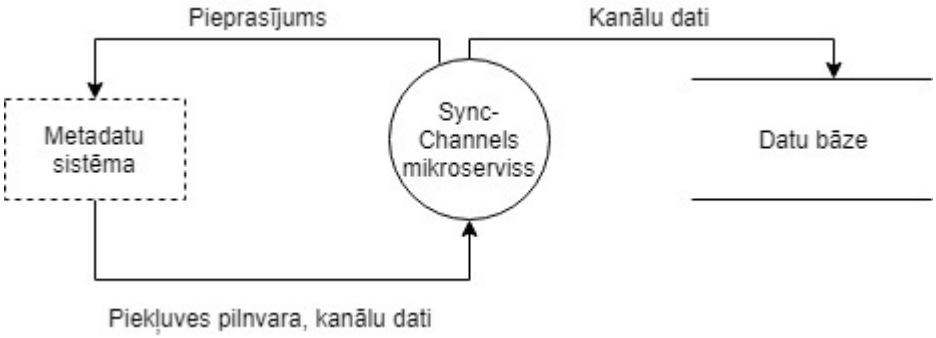
2.2.6. att. **Datu plūsmas diagramma 1. līmenis – SyncEpg**

2.2.7. 2. līmenis – SyncEpg



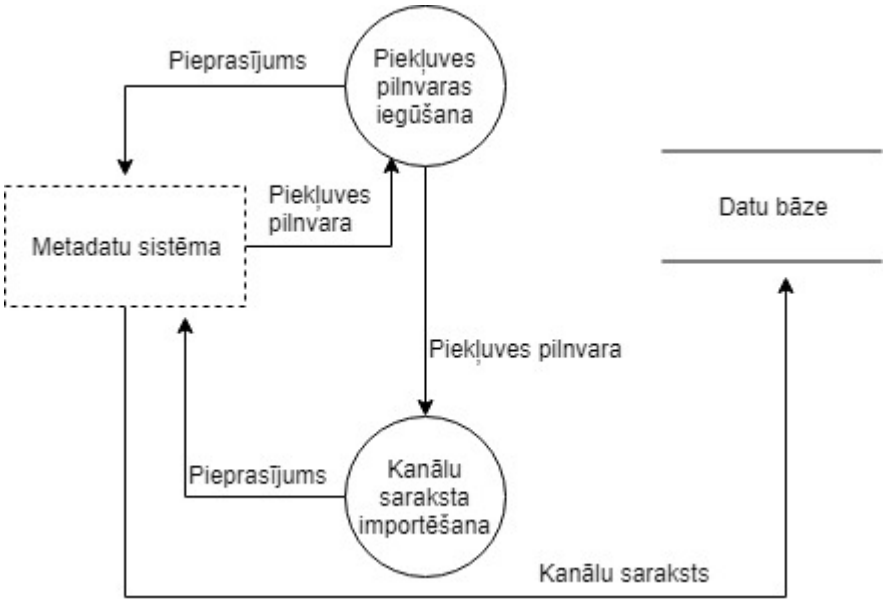
2.2.7. att. **Datu plūsmas diagramma 2. līmenis – SyncEpg**

2.2.8. 1. līmenis – SyncChannels



2.2.8. att. **Datu plūsmas diagramma 1. līmenis – SyncChannels**

2.2.9. 2. līmenis – SyncChannels



2.2.9. att. **Datu plūsmas diagramma 2. līmenis – SyncChannels**

2.3. Datu bāzes projektējums

Sistēmas izveidošanā un uzturēšanā tiek izmantota Cassandra datu bāze, kas ir atvērtā pirmkoda NoSQL izplatīta datu bāzes pārvaldības sistēma. Tā nodrošina mērogojamību, jo ir dalīta sistēma, kā arī tā ir pieejama kā atvērtā pirmkoda programmatūra. Katram mikroservisam ir sava datu bāze ar tabulām.

2.3.1. Datu bāzes tabulu projektējums

Datu bāzes tabulu aprakstā katra no tabulām tiek aprakstīta, izmantojot šādu struktūru:

- Lauka nosaukums
- Datu tips
- Apraksts
- Piezīmes

Saīsinājumi piezīmju laukā:

- PK - Primary Key (primārā atslēga)
- PTK – Partition Key (partīcijas atslēga)

- CLK – Clustering Key (grupēšanas atslēga)

Katrai tabulai ir unikāla primārā atslēga, kas var sastāvēt no vairākiem laukiem, kuri var saturēt partīcijas, grupēšanas atslēgas.

Vienam lietotājam var būt vairāki profili, tāpēc tabulās tiek glabāts lietotāja konkrētā profila, ar ko viņš ir veicis darbības, identifikators.

2.3.4.1 Tabula "user_like_dislike"

Tabulā tiek glabāta informācija par to, kādu vērtējumu ir ievietojis lietotājs konkrētam video.

2.3.4.1. tabula
Datu bāzes tabulas "user_like_dislike" projektējums

Lauka nosaukums	Datu tips	Apraksts	Piezīmes
profile_id	int	Profila identifikators	PTK
vod_id	bigint	Video identifikators	PTK
is_liked	boolean	Lauks, kas identificē vērtējuma 'patīk' aktivitāti	
is_disliked	boolean	Lauks, kas identificē vērtējuma 'nepatīk' aktivitāti	

2.3.4.2 Tabula "vod_like_dislike"

Tabulā tiek glabāta informācija par video vērtējumu skaitu.

2.3.4.2. tabula
Datu bāzes tabulas "vod_like_dislike" projektējums

Lauka nosaukums	Datu tips	Apraksts	Piezīmes
vod_id	bigint	Video identifikators	PTK
is_liked	counter	Lauks, kas fiksē vērtējuma 'patīk' skaitu	
is_disliked	counter	Lauks, kas fiksē vērtējuma 'nepatīk' skaitu	

2.3.4.3. Tabula “view_user_epg”

Tabulā tiek glabāta informācija par arhīva ieraksta skatījumu.

2.3.4.3. tabula
Datu bāzes tabulas “view_user_epg” projektējums

Lauka nosaukums	Datu tips	Apraksts	Piezīmes
profile_id	int	Profila identifikators	PTK
item_type	varchar	Arhīva ieraksta veids	PTK
item_id	bigint	Arhīva ieraksta identifikators	PTK
created	timestamp	Ieraksta izveidošanas laika zīmogs	

2.3.4.4. Tabula “view_user_vod”

Tabulā tiek glabāta informācija par video skatījumu.

2.3.4.4. tabula
Datu bāzes tabulas “view_user_vod” projektējums

Lauka nosaukums	Datu tips	Apraksts	Piezīmes
profile_id	int	Profila identifikators	PTK
item_type	varchar	Video veids	PTK
item_id	bigint	Video identifikators	PTK
created	timestamp	Ieraksta izveidošanas laika zīmogs	

2.3.4.5. Tabula “epg”

Tabulā tiek glabāta informācija par arhīva ierakstiem.

2.3.4.5. tabula
Datu bāzes tabulas “epg” projektējums

Lauka nosaukums	Datu tips	Apraksts	Piezīmes
broadcast_id	bigint	Pārraides identifikators	PTK
lang	text	Valodas kods	CLK
next_broadcast_id	bigint	Nākamās pārraides identifikators	
channel_id	int	Kanāla identifikators	
channel_name	text	Kanāla nosaukums	
content_type	text	Satura veids	

series_original_title	text	Seriāla oriģinālais nosaukums	
movie_original_title	text	Filmas oriģinālais nosaukums	
series_title	text	Seriāla nosaukums	
movie_title	text	Filmas nosaukums	
tv_show_title	text	Raidījuma nosaukums	
category	text	Pārraides kategorija	
consumer_rule_recordable	text	Atļauja klientam ierakstīt pārraidi	
consumer_rule_restartable	text	Atļauja klientam restartēt pārraidi	
consumer_rule_pauselive	text	Atļauja klientam apstādināt tiešraidi	
consumer_rule_startover	text	Atļauja klientam sākt pārraidi no sākuma	
start_time	timestamp	Pārraides sākšanās laiks	
end_time	timestamp	Pārraides beigšanās laiks	
censure	int	Pārraides cenzūra	
content_id	int	Satura identifikators	
genres	set<text>	Pārraide žanri	
series_annotation	text	Seriāla anotācija	
movie_annotation	text	Filmas anotācija	
directors	set<text>	Pārraides režisori	
actors	set<text>	Pārraides aktieri	
year	int	Pārraides gads	
series_id	int	Seriāla identifikators	
series_name	text	Seriāla nosaukums	
episode_name	text	Epizodes nosaukums	
season_nr	int	Sezonas numurs	
episode_nr	int	Epizodes numurs	
episode_annotation	text	Epizodes anotācija	
picture_path	text	Attēla ceļš	

Datu tips set<text> satur elemetu grupu ar unikālām vērtībām, šis datu tips tika izvēlēts, lai varētu vienā laukā glabāt vairākas vērtības, piemērām, laukā "actors" tiek glabāti vairāki aktieri.

2.3.4.6. Tabula "channels"

Tabulā tiek glabāta informācija par kanāliem.

2.3.4.6. tabula

Datu bāzes tabulas "channels" projektējums

Lauka nosaukums	Datu tips	Apraksts	Piezīmes
channel_id	int	Kanāla identifikators	PTK
lang	text	Valodas kods	CLK
title	text	Kanāla nosaukums	
description	text	Kanāla apraksts	
short_title	text	Kanāla īsais nosaukums	
channel_number	int	Kanāla numurs	
url	text	Kanāla adrese internetā	
aspect_ratio	text	Attēla attiecība	
logo_picture_url	text	Logo bildes adrese internetā	
logo_picture_file_name	text	Logo bildes faila nosaukums	
logo_picture_aspect_ratio	text	Logo bildes attiecība	
logo_picture_width	int	Logo bildes platums	
logo_picture_height	int	Logo bildes augstums	
lead_picture_url	text	Otra logo bildes adrese internetā	
lead_picture_file_name	text	Otra logo bildes faila nosaukums	
lead_picture_aspect_ratio	text	Otra logo bildes attiecība	
lead_picture_width	int	Otra logo bildes platums	
lead_picture_height	int	Otra logo bildes augstums	
languages	set<text>	Kanāla valodas	
subtitles	text	Kanāla subtitri	
genres	text	Kanāla žanri	
bitrates	set<text>	Bitu pārraides ātrums	
stream_id	text	Pārraides identifikators	

stream_platform	text	Pārraides platforma	
stream_languages	set<text>	Pārraides valodas	
stream_subtitles	text	Pārraides subtitri	

Datu tips set<text> satur elementu grupu ar unikālām vērtībām, šis datu tips tika izvēlēts, lai varētu vienā laukā glabāt vairākas vērtības.

3. TESTĒŠANAS DOKUMENTĀCIJA

3.1. Ievads

Šajā nodaļā tiek aprakstīta izstrādāto mikroservisu testēšana un tās rezultāti. Bija nepieciešams testēt tīmekļa servisus, kuri ir veidoti pēc REST arhitektūras stila, tāpēc vienojoties ar citiem izstrādātājiem, tika nolemts, ka tiks veidoti API testi. Automatizētie testi tika izveidoti, izmantojot *Codeception* automatizētās testēšanas ietvaru [3]. Lai pārliecinātos par atbilstību programmatūras prasību specifikācijai, tad testus veica pats izstrādātājs.

3.2. Testpiemēru specifikācija

Veiktie testi aprakstīti tabulas formā. Apkopoti veiktie automatizētie testi un manuālie testi. Tie tiek aprakstīti tabulas formā:

- Nr. – testa numurs pēc kārtas;
- Testa apraksts;
- Sagaidāmais rezultāts;
- Rezultāts (‘+’, ja rezultāts sakrīt ar gaidāmo rezultātu, citādi tiek aprakstīta kļūdas iemesls).

3.2.1. LikeDislike

Testi tiek pildīti ar derīgu piekļuves pilnvaru.

Nr.	Testa apraksts	Sagaidāmais rezultāts	Rezultāts
1.	Tiek izsaukta funkcija, kas nodrošina ‘patīk’ vērtējuma ievietošanu.	HTTP atbildes kods ir 200.	+
2.	Tiek izsaukta funkcija, kas nodrošina ‘nepatīk’ vērtējuma ievietošanu.	HTTP atbildes kods ir 200.	+
3.	Tiek izsaukta funkcija, kas nodrošina ievietoto vērtējumu atgriešanu.	HTTP atbildes kods ir 200.	+
4.	Tiek pārbaudīts, vai atgrieztie dati sakrīt ar JSON tipu.	HTTP atbildes kods ir 200.	+
5.	Tiek pārbaudīts, vai atgrieztie dati satur JSON vērtības.	HTTP atbildes kods ir 200.	+

Testi tiek pildīti bez piekļuves pilnvaras, lai pārliecinātos, ka neautorizējies lietotājs nevar izmantot izveidoto funkcionalitāti.

Nr.	Testa apraksts	Sagaidāmais rezultāts	Rezultāts
1.	Tiek izsaukta funkcija, kas nodrošina ‘patīk’ vērtējuma ievietošanu.	HTTP atbildes kods ir 401.	+
2.	Tiek izsaukta funkcija, kas nodrošina ‘nepatīk’ vērtējuma ievietošanu.	HTTP atbildes kods ir 401.	+
3.	Tiek izsaukta funkcija, kas nodrošina ievietoto vērtējumu atgriešanu.	HTTP atbildes kods ir 401.	+

3.2.2. View

Testi tiek pildīti ar derīgu piekļuves pilnvaru.

Nr.	Testa apraksts	Sagaidāmais rezultāts	Rezultāts
1.	Tiek izsaukta funkcija, kas nodrošina arhīva ieraksta skatījuma reģistrēšanu.	HTTP atbildes kods ir 200.	+
2.	Tiek izsaukta funkcija, kas nodrošina video skatījuma reģistrēšanu.	HTTP atbildes kods ir 200.	+

Testi tiek pildīti bez piekļuves pilnvaras, lai pārliecinātos, ka neautorizējies lietotājs nevar izmantot izveidoto funkcionalitāti.

Nr.	Testa apraksts	Sagaidāmais rezultāts	Rezultāts
1.	Tiek izsaukta funkcija, kas nodrošina arhīva ieraksta skatījuma reģistrēšanu.	HTTP atbildes kods ir 401.	+
2.	Tiek izsaukta funkcija, kas nodrošina video skatījuma reģistrēšanu.	HTTP atbildes kods ir 401.	+

3.2.3. SyncEpg

Nr.	Testa apraksts	Sagaidāmais rezultāts	Rezultāts
1.	Tiek pārbaudīts, vai ir iegūta meta datu sistēmas piekļuves pilnvara	Piekļuves pilnvara ir iegūta, un tā ir derīga.	+
2.	Tiek pārbaudīts, vai iegūto datu lauki no meta datu sistēmas atbilst datu bāzē izveidotajai tabulai.	Datu lauki sakrīt ar datu bāze izveidotajiem laukiem.	+
3.	Tiek pārbaudīts, vai datu bāze ir ievietoti saņemtie dati.	Datu bāzē ir importēti atbilstošā kanāla arhīva ieraksti.	+

3.2.4. SyncChannels

Nr.	Testa apraksts	Sagaidāmais rezultāts	Rezultāts
1.	Tiek pārbaudīts, vai ir iegūta meta datu sistēmas piekļuves pilnvara	Piekļuves pilnvara ir iegūta, un tā ir derīga.	+
2.	Tiek pārbaudīts, vai iegūto datu lauki no meta datu sistēmas atbilst datu bāzē izveidotajai tabulai.	Datu lauki sakrīt ar datu bāze izveidotajiem laukiem.	+
3.	Tiek pārbaudīts, vai datu bāze ir ievietoti saņemtie dati.	Datu bāzē ir importēti kanālu saraksta dati.	+

4. PROJEKTA ORGANIZĀCIJA

Darbs tika izstrādāts pēc programmatūras izstrādes metodoloģijas Scrum, jo projekta sākumā bija zināms sagaidāmais rezultāts, taču visas detaļas un prasības nebija precīzi definētas, un tās laika gaitā varēja mainīties. Projekts tika sadalīts vairākās mazākās daļās – mikroservisos. Pirms katra mikroservisa izstrādes tika veiktas pārrunas ar projekta vadītāju, lai apzinātu mikroservisa sagaidāmo darbību, precizētu un saskaņotu prasību specifikāciju. Tika izveidots darāmo darbu saraksts, lai efektīvi varētu uzbūvēt jaunu servisu. Paveicamo darbu saraksts tika reģistrēts *Team Foundation Server* sistēmā. Katru dienu notika pārrunas ar mikroservisu projektu vadītāju, lai apspriestu izdarītos darbus, problēmas.

Mikroservisu izstrādes projekts ir daļa no lielāka projekta, tāpēc katru dienu notika visas platformas izstrādes darbinieku *StandUp* (15 minūšu gara sapulce, kuras laikā katrs izklāstīja savus paveiktos un plānotos darbus, kā arī kādas ir aizķeršanās, ja tādas ir). Lielākā projekta organizācija un darbi tika sadalīti iterācijās jeb sprintos. Viena sprinta ilgums bija divas nedēļas. Autore piedalījās no 79. līdz 84. sprintam.

5. KVALITĀTES NODROŠINĀŠANA

Lai izstrādātajam projektam nodrošinātu pēc iespējas augstāku kvalitāti, darba izstrādes laikā tika veidots programmatūras projektējums, pēc tam arī testēšanas dokumentācija.

Programmējot tika ievēroti labās prakses pamatprincipi, kā koda izkārtojums, ieturēts vienots stils atbilstošu mainīgo veidošanā, lai kodu varētu viegli lasīt. Pēc katras funkcionalitātes pabeigšanas kods tika nodots mentoram caur *GitLab* izskatīšanai. Ja kodā tika atrastas, kādas kļūdas, tad tas tika nodots atpakaļ izstrādātājam, lai tās izlabotu. Tikai tad, kad viss atbilda prasībām, izveidotais kods tika sapludināts ar kopējo izstrādes zaru.

Lai rūpīgi sekotu līdzi projekta izstrādes gaitai un neaizkavētos pie radušajām problēmām, kā minēts iepriekšējā nodaļā, tika veiktas sapulces un pārrunas ar projekta vadītāju un citiem izstrādātājiem.

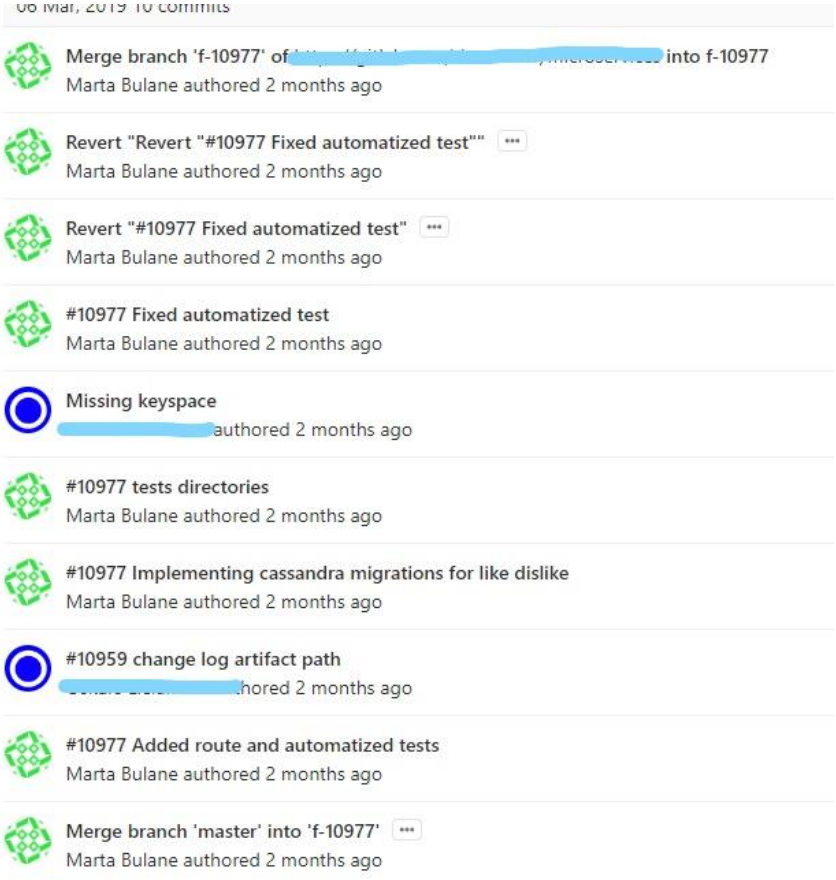
Pēc katras funkcionalitātes izveidošanas, lai pārliecinātos par korektu programmatūras darbību, tika izveidoti un veikti REST API testi. Ja tika atrastas kādas kļūdas, izstrādātājs tās izlaboja un atkal veica testēšanu.

6. KONFIGURĀCIJU PĀRVALDĪBA

Sākumā programmatūras pirmkods tika saglabāts lokāli, un pēc tam caur *SourceTree,* kas ir bezmaksas *Git* klients izstrādātājiem uz Windows operētājsistēmas, tas tika uzglabāts, izmantojot versiju kontroles sistēmu *Git*, *GitLab* sistēmā attiecīgajā repozitorijā. Tādējādi, ja ir radušās kādas kļūdas un ir nepieciešams atgriezt iepriekšējo versiju, to ir viegli izdarīt, kā arī pateicoties šim izmantotajam pakalpojumam, citi izstrādātāji ir spējīgi sekot līdzi projekta gaitai. Attēlā (*6. att.*) var redzēt saglabāto versiju vēsturi.

Pirms katra jauna uzdevuma sākšanas, tika izveidots jauns zars, kura nosaukuma veidošanā tika ieturēts vienots stils, piemēram, **f-12345-updated-database-tables**, pēc shēmas **type-ID**, kur **f** nozīme "*feature*", bet **12345** darba identifikatoru. Lai jauno versiju pievienotu *GitLab* sistēmā, un vēlāk to sapludinātu ar galveno izstrādes zaru, tika veidoti sapludināšanas pieprasījumi (*angl*. merge requests).

Lai izveidotos servisus būtu iespējams uzstādīt vairākas vidēs un izveidot savienojumus ar datu bāzēm un citām sistēmām, nemainot programmatūras pirmkodu, tika izmantoti vides mainīgie (*angl*. environment variables).



6. att. **Git commit vēstures skats GitLab**

7. DARBIETILPĪBAS NOVĒRTĒJUMS

Pirms katra mikroservisa izstrādes prakses vadītājs veica katra servisa darbietilpības novērtējumu funkcijpunktu veidā, balstoties uz savu un darba autores pieredzi. Viens funkcijpunkts atbilsts 8 stundām jeb vienai persondienai. Pēc projekta izstrādes izveidotais darbietilpības novērtējums tika salīdzināts ar reālo darbietilpības novērtējumu.

7. tabula

Darbietilpības novērtējums funkcijpunktos

Darba nosaukums	**Darbietilpības novērtējums funkcijpunktos**	**Reālā darbietilpība funkcijpunktos**
Darba tehnoloģiju apgūšana	7	8
Mikroservisu izstrādei darāmo darbu plānošana	5	5
LikeDislike funkcionalitāte	11	13
View funkcionalitāte	8	5
SyncEpg funkcionalitāte	14	19
SyncChannels funkcionalitāte	5	5
Testēšana	7	8
Kopā:	57	63

Pirms darba izstrādes darbietilpības novērtējums funkcijpunktos bija 57 persondienas, taču realitātē projekta izstrādei tika patērētās 63 persondienas, kas sanāk 3 personmēneši. Patērētais laiks katru dienu tika ievadīts uzņēmuma darba laika uzskaites sistēmā, pēc tā arī tika veidota reālā darbietilpība. Izteiktā prognoze tika pārsniegta par 6 persondienām, jo autores darba iemaņas ar izstrādē izmantoto programmēšanas valodu PHP, kā arī ar citām darbā izmantotajām tehnoloģijām, bija minimālas. Ārējās sistēmas tehnisku problēmu dēļ, kas bija nepieciešama izstrādājot *SyncEpg* un *SyncChannels* mikroservisus, darbs aizkavējās par vairākām persondienām. Lielu daļu servisu veidošanā aizņēma kļūdu labošana. *View* mikroservisam projekta izstrādes sākumā bija paredzēta plašāka funkcionalitāte, tomēr projekta izstrādes gaitā, projekta vadītājs nolēma funkcionalitāti sašaurināt, tāpēc servisa darbietilpības novērtējums ir lielāks nekā reālā darbietilpība.

8. SECINĀJUMI

Kvalifikācijas darba ietvaros tika izstrādāti mikroservisi izklaides platformas sistēmas servera pusē. Paralēli izveidotajai programmatūrai, tika izstrādāta atbilstoša dokumentācija, kas sastāv no prasību specifikācijas, programmatūras projektējuma apraksta un testēšanas dokumentācijas.

Darba autores priekšzināšanas par projektā izmantotajām tehnoloģijām bija ļoti minimālas, izstrādes gaitā tika iegūta nozīmīga pieredze un jaunas zināšanas ar šādām tehnoloģijām:

- programmatūras izstrādes rīki: Visual Studio Code, TablePlus, Postman;
- programmēšanas valoda: PHP;
- datu bāzu vadība: Cassandra;
- programmatūras izstrādes ietvars: Lumen;
- testēšanas ietvars: Codeception;
- virtualizācijas platforma: Docker;
- programmatūras konfigurācijas vadība: Microsoft Team Foundation Server 2018, GitLab, versiju kontroles sistēma Git, SourceTree;
- izstrādes process: programmatūras izstrādes metodoloģija: Scrum.

Projekta izstrādes gaitā dažas prasības tika mainītas un papildinātas, radās tehniskas problēmas, tas uzlaboja autores spēju pielāgoties, kas ir svarīga piedaloties jebkāda veida projektos.

Pateicoties tam, ka tika izstrādāts reāls produkts, kas būs pieejams platformas lietotājiem, tika iegūta milzīga pieredze strādāt un sadarboties komandā, kā arī deva vērtīgu ieskatu mūsdienu programmatūras izstrādē.

ATSAUCES

1. PHP dokumentācija [tiešsaiste] – [atsauce 24.05.2019]. Pieejams: https://php.net/
2. Akadēmiskā terminu datu bāze [tiešsaiste] – [atsauce 15.05.2019]. Pieejams: http://termini.lza.lv/
3. Codeception dokumentācija [tiešsaiste] – [atsauce 02.05.2019]. Pieejams: https://codeception.com/
4. Cassandra dokumentācija [tiešsaiste] – [atsauce 11.05.2019]. Pieejams: http://cassandra.apache.org/
5. "Microservices architecture" [tiešsaiste] – [atsauce 30.04.2019]. Pieejams: https://microservices.io/

PIELIKUMI

Koda fragmenti

LikeDislike vērtējuma pievienošana

```
//Funkcija nodrošina vērtējuma ievietošanu.
public function set(Request $request, $id)
    {
        $validator = Validator::make($request->all(), [
            'is-liked' => 'required_without_all:is-disliked',
            'is-disliked' => 'required_without_all:is-liked',

        ]);

        if ($validator->fails()) {
            $errorCode = ErrorCode::MISSING_PARAMS;
            $details = $validator->messages()->first();
            return $this->getErrorResponse($errorCode, $details);
        }

        $item = [
            'profile-id' => (int) $request->credentials->profile,
            'vod-id' => new Cassandra\Bigint($id),
            'is-liked' => $request->input('is-liked'),
            'is-disliked' => $request->input('is-disliked'),
        ];

        $likeDislike = new LikeDislikeUser;
        $checkIfExists = $likeDislike->get($item);
        if ($checkIfExists) {
            $this->deleteUserLikeDislike($checkIfExists);
        }

        $likeDislike = new LikeDislikeUser;
        $likeDislikeVod = new LikeDislikeVod;

        $like = $item['is-liked'];
        $dislike = $item['is-disliked'];

        if ($like == 'true') {
            $item['is-liked'] = true;
            $likeDislike->insertUser($item);
            $likeDislikeVod->addLike($item);
        }

        if ($dislike == 'true') {
            $item['is-disliked'] = true;
            $likeDislike->insertUser($item);
            $likeDislikeVod->addDislike($item);
        }

        if ($like == 'false' || $dislike == 'false') {
            $likeDislike->delete($item);
        }

        return '';
```

```
    }
```

LikeDislike vērtējuma atgriešana

```
//Funkcija atgriež vērtējumus.
public function get(Request $request, $id)
    {
        $item = [
            'profile-id' => (int) $request->credentials->profile,
            'vod-id' => new Cassandra\Bigint($id),
        ];

        $likeDislikeUser = new LikeDislikeUser();
        $likeDislikeUser->get($item);

        $likeDislikeVod = new LikeDislikeVod();
        $likeDislikeVod->get($item);

        $likeDislike = new LikeDislike($likeDislikeUser, $likeDislikeVod);

        if ($likeDislike) {
            $response = $this->getResponse($likeDislike);
        }
        if (!$likeDislike) {
            $response = $this->getNullResponse();
        }

        return $response;
    }



//Funkcija no datu bāzes atgriež pieprasītos datus.
public function get($item)
    {
        $query = "SELECT * FROM " . env('CASSANDRA_DATABASE_KEYSPACE') .
".user_like_dislike
                    WHERE profile_id = ? AND vod_id = ? ";
        $statement = $this->session->prepare($query);
        $result = $this->session->execute($statement, array('arguments' =>
array($item['profile-id'], $item['vod-id'])));
        foreach ($result as $item) {
            $this->profile_id = $item['profile_id'];
            $this->vod_id = $item['vod_id'];
            $this->is_liked = $item['is_liked'];
            $this->is_disliked = $item['is_disliked'];
            return $this;
        }
    }
```

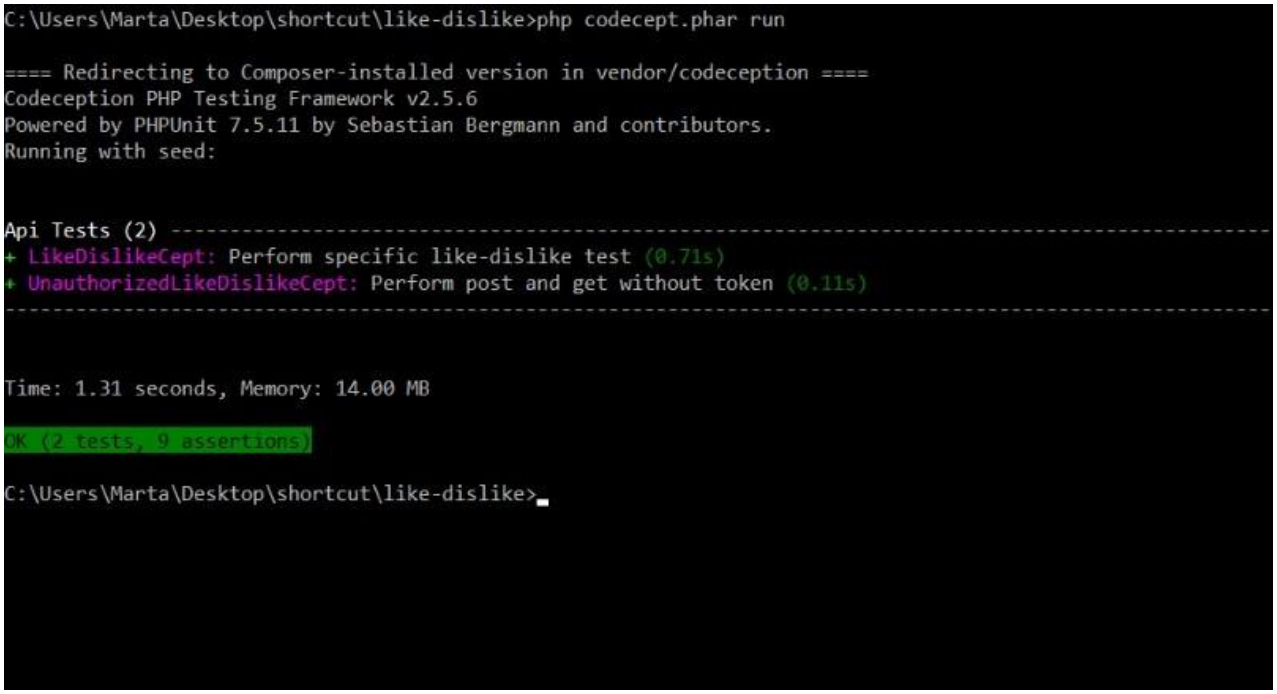
SyncEpg un SyncChannels palīgfunkcija

```
//Funkcija izveido cURL savienojumu ar ārējo sistēmu un saņem datus.
public static function getDataFromExternalSource($url)
    {
        $userName = getenv('MDS_USERNAME');
        $password = getenv('MDS_PASSWORD');
        $metaDataAuthUrl = getenv('MDS_HOST_URL') . getenv('MDS_AUTH');
        $postData = array(
            "username" => $userName,
            "password" => $password,
```

```
        );

        $postString = json_encode($postData);
        $params = array(
            CURLOPT_HTTPHEADER => array('Content-Type: application/json',
'Connection: keep-alive', 'Content-Length: ' . strlen($postString)),
            CURLOPT_CUSTOMREQUEST => "POST",
            CURLOPT_POSTFIELDS => $postString,
        );
        $token = null;
        $tokenData = self::sendCustomCurl($metaDataAuthUrl,
$params)['response'];
        if (isset(json_decode($tokenData)->token)) {
            $token = json_decode($tokenData)->token;
        }

        $header = array();
        $header[] = 'Authorization: Bearer ' . $token;

        $curl = curl_init(getenv('MDS_HOST_URL') . getenv('MDS_EPG'));

        curl_setopt_array($curl, array(
            CURLOPT_URL => $url,
            CURLOPT_USERAGENT => 'curl/shortcut-microservices',
            CURLOPT_FOLLOWLOCATION => true,
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_ENCODING => 'UTF-8',
            CURLOPT_SSL_VERIFYHOST => 0,
            CURLOPT_SSL_VERIFYPEER => 0,
            CURLOPT_HTTPHEADER => $header,
        ));

        $result = curl_exec($curl);
        $httpCode = curl_getinfo($curl, CURLINFO_HTTP_CODE);
        $error = curl_error($curl);
        if ($error != "" || $httpCode != "200") {
            $result = "";
        }
        curl_close($curl);
        return $result;
    }

//Funkcija izveido cURL savienojumu ar ārejo sistēmu, lai iegūtu piekļuves
pilnvaru.
    public static function sendCustomCurl($url, $params = null)
    {
        $curl = curl_init();
        curl_setopt_array($curl, array(
            CURLOPT_URL => $url,
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_HEADER => false,
            CURLOPT_SSL_VERIFYPEER => false,
        ));
        if ($params != null) {
            foreach ($params as $key => $value) {
                curl_setopt($curl, $key, $value);
            }
        }
        $response = curl_exec($curl);
        $error = curl_error($curl);
        curl_close($curl);
        return array('response' => $response, 'error' => $error);
    }
```

Testēšanas rezultāti piemēri

LikeDislike REST API testi



Kvalifikācijas darbs „***Mikroservisu izstrāde izklaides platformai aizmugurējā sistēmā***” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: ***Marta Bulāne*** _____ **___.05.2019**.

Rekomendēju darbu aizstāvēšanai
Darba vadītājs: ***M. dat. Oskars Ozols*** _____ **___.05.2019.**

Recenzents: ***Uldis Karlovs-Karlovskis***

Darbs iesniegts 27.05.2019.
Kvalifikācijas darbu pārbaudījumu komisijas sekretāre: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2019. prot. Nr. _____

Komisijas sekretārs(-e): _____