LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

# Sociālo mediju ziņojumu plūsmu pārskata vietne

KVALIFIKĀCIJAS DARBS

Autors: Rihards Gailis

Studenta apliecības Nr: rg14009

Darba vadītājs: Mg.sc. ing Toms Mediņš

RĪGA 2016

# ANOTĀCIJA

Kvalifikācijas darba “Sociālo mediju ziņojumu plūsmu pārskata vietne” mērķis ir izstrādāt tīmekļa vietni, kura spētu palīdzēt, piemēram, kāda pasākuma organizatoram sekot līdzi cilvēku attieksmei un attiecīgi veikt izmaiņas pašreizējā vai nākamajos pasākumos, lai uzlabotu cilvēku labsajūtu un pieredzi. Šī tīmekļa vietne ir pieejama caur tīmekļa pārlūku jebkuram interesantam, taču administratora pieeja tiek atļauta tikai kādai iepriekš definētai cilvēku grupai. Tīmekļa vietnes administrators spēs izveidot jaunu mediju sienu, kurā tiks parādītas dažādas cilvēku sīkziņas atkarībā no administratora izvēlētajiem parametriem sienas izveidošanas brīdī. Sistēmas jebkuram lietotājiem būs iespējams apskatīt tikai administratoru izveidotās mediju sienas.

Programmatūras nepieciešamā funkcionalitāte izstrādāta izmantojot *Java*, kas tiek izmantota servera puses izveidei un tehnoloģijas kā *JS, HTML, CSS* tiek izmantotas klienta puses izveidei. Izstrādāto projektu ievieto *HANA Cloud* platformā.

Atslēgvārdi: Java, HTML, HANA Cloud Platforma, CSS, JS.

# ABSTRACT

Aim of "Social media news feed overview website" qualification work is to develop a web site that could help, for example, an event organizer to keep track of people's attitudes and to do adjustments to improve people's well-being and experience for future events. This web site is accessible through any web browser for anyone who is interested, however the administrator access is only allowed to a pre-defined group of people. Web site administrator will be able to create a new media wall, which will display a variety of human Tweets, depending on the administrator's selected parameters at the time of creation of the wall. All users of the system will be able to view media walls created by administrator.

Required functionality for software development was built using *Java*, which was used for server-side development and technologies as *JS*, *HTML*, *CSS* were used for client-side development. Project  is deployed on *HANA Cloud* platform.



Keywords: Java, HTML, HANA Cloud Platform, CSS, JS.

# SATURS

# PAMATDEFINĪCIJAS

**Administrators** – autorizēts lietotājs, kuram piešķirtas sistēmas pārvaldes tiesības.

**JSON** – *(JavaScript Object Notation - JavaScript Objektu Notācija)* — teksta un datu apmaiņas formāts, kas tiek izmantots, lai pārsūtītu strukturētus datus, parasti Ajax Web lietojumprogrammas.

**PPS** – Programmatūras prasību specifikācija.

**PPA** – Programmatūras projektējuma apraksts.

**SVN (Subversion)** – atvērtā koda versiju kontroles sistēma.

**Java** – Objektorientēta programmēšanas valoda.

**JavaScript** (*JS*)– klienta puses skriptu valoda, ko izmanto tīmekļu vietņu izstrādē.

**HTML** *(Hypertext Markup Language)* - standartizēta teksta failu iezīmēšanas valoda, kas paredzēta tīmekļa vietņu veidošanai.

**HTTP** *(Hyper Text Transfer Protocol)* – Tīkla protokols, kas nodrošina informācijas apmaiņu globālajā tīmeklī.

**CSS** *(Cascading Style Sheets)* - Valodas HTML papildu iespēja, kas ļauj globālā tīmekļa pakalpojumu izstrādātājiem un tīkla lietotājiem veidot īpašus lappušu izklājumus.

**API** – iepriekš definētu klašu, procedūru, funkciju, struktūru un konstanšu kopums, kas tiek pasniegts kā pielikums (bibliotēkas, servisi), kuru iespējams izmantot ārējiem programmatūras produktiem.

**RESTful API –** lietojumprogrammu interfeiss, kas izmanto HTTP pieprasījumus - GET, POST, PUT un DELETE, lai varētu veikt darbības ar datiem.

**UTF-8** – Teksta kodēšanas metode, ko izmanto rakstzīmju un citu simbolu attēlošanai unikoda standartā.

**CRUD** – (**C**reate, **R**ead, **U**pdate, **D**elete) Akronīms, kurš apzīmē darbības ar datiem, četras
sistēmas funkcijas – izveidot, lasīt, atjaunināt, dzēst.

**Spring framework** – **(**Spring satvars) - atvērtā koda Java platforma, kuru var izmantot jebkurā java programmu izveidē. Satvara galvenais mērķis ir atvieglot pašu Java programmmu izstrādi.

**PostgreSQL** – objektu relāciju datu bāzes pārvaldības sistēma.

**Maven –** programmatūras projektu pārvadības rīks. Pamatojoties uz konceptu par projekta objekta modeli (*POM*), ar Maven palīdzību var buvēt projektu un saņemt kļudas paziņojumus.

**SAP HCP** *(SAP HANA Cloud Platform)* **-** SAP platforma kā serviss (*PaaS*) mākoņdatošanas seviss, uz kura tiek izvietota izstrādātā sistēma.

**MongoDB** – atvērtā avota datu bāze, kas izmanto dokumentu orientētu datu modeli.

**SAP HANA DB** – brīvpiekļuves atmiņā esoša relāciju datu bāze, kas reālā laikā spej apstrādāt lielus atmiņas datus.

**Apache Tomcat –** atvērtā pirmkoda tīmekļa serveris un
Java serversīklietotnes konteiners.

**Attlassian JIRA** – kļūdu izsekošanas, problēmu izsekošanas un projektu pārvaldības rīks.

**Java Persistence API**(*JPA*) – *Java* specifikācija priekš datu piekļuves un datu pārvaldības starp *Java* objektu un relāciju datubāzi.

# IEVADS

Kvalifikācijas darbā tiek izstrādāts projekts “Sociālo mediju ziņojumu plūsmu pārskata vietne”. Šī tīmekļa vietne ir veidota kā ātra informācijas uzziņas vietne priekš specifiskas informācijas iegūšanas, jo mūsdienās palielinoties datu apjomam internetā ir grūti izfiltrēt no simtiem ziņojumu tieši tos, kuri būs vajadzīgi, tāpēc programma lietotāja vietā izfiltrē viņam vēlamās cilvēku sīkziņas un pēc viņa padotajiem parametriem atgriež tās vienā skatā.

Ar iztrādātajām funkcijām, kurām piekļuve ir tikai administratoram, jeb autentificētam lietotājam ir iespējams izveidot jaunu sociālo mediju ziņojumu sienu, atjaunot to un dzēst. Ar administratora attiecīgi padotajiem parametriem mediju sienā tiek parādītas dažādu cilvēku sīkziņas, kuras ir iegūtas no sociālo mediju vietnēm. Katras mediju sienas sīkziņas meklēšanas parametrus ir iespējams rediģet, lai iegūtu sev vēlamāku un precīzāku informāciju.

Tīmekļa vietne tiek izstrādāta, izmantojot programmēšanas valodu Java, kā arī lapas dizaina un informācijas attēlošanai tiek izmantots Javascript, HTML un CSS. Dati izstrādes vidē lokāli tiek glabāti PostgreSQL un MongoDB datu bāzēs, bet kad sistēma tiks palaista tīklā, tad dati tiks glabāti SAP HANA datubāzē.

# 1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

## 1.1. Ievads

### 1.1.1. Nolūks

Šīs programmatūras prasību specifikācijas (PPS) nolūks ir aprakstīt "Sociālo mediju ziņojumu plūsmu pārskata vietne" prasības un funkcionalitāti. Dokuments paredzēts lietotājiem un izstrādātājiem, lai nodrošinātu kvalitatīvu programmatūras izstrādi.

### 1.1.2. Darbības sfēra

"Sociālo mediju ziņojumu plūsmu pārskata vietne" ir paredzēta, lai uzlabotu pasākumu, sanāksmju un konferenču efektivitāti, kvalitāti un pieredzi to apmeklētājiem, jo atsaucoties uz cilvēku sīkziņām sociālajos medijos attiecīgi var veikt izmaiņas, kas ne tikai uzlabo, bet arī atvieglo organizatora darbu. Par piemēru var minēt to, ka ar šīs tīmekļa vietnes palīdzību organizatoram nav jāmokās ar lielu informācijas apjomu, tā vietā viņš var izmantot šīs tīmekļa vietnes funkcijas, lai tā atgrieztu tikai nepieciešamo informāciju vienā mājaslapas skatā.

### 1.1.3. Dokumenta pārskats

Programmatūras prasību specifikācija sastāv no četrām apakšnodaļām:

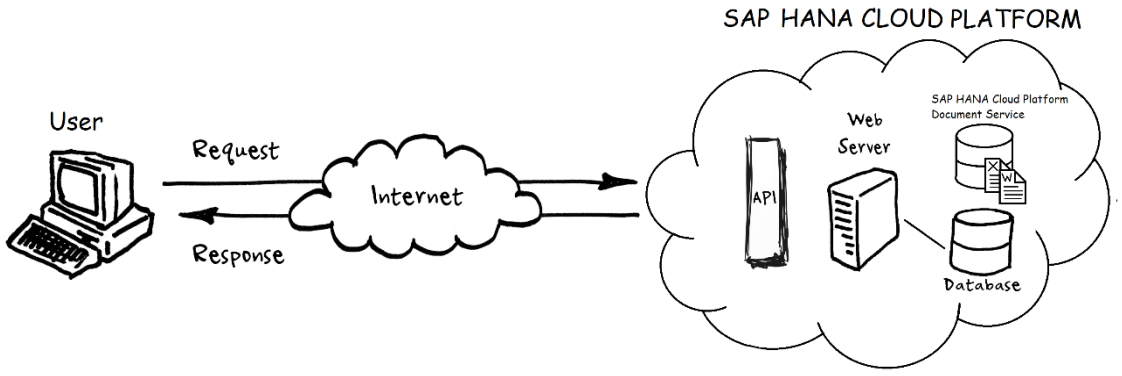
1. Ievads - raksturo dokumentu, tā mērķi, kādēļ tas tiek rakstīts.
2. Vispārējs apraksts - vispārīga projekta apraksts, kur iekļaujas produkta funkciju apraksts, sistēmas ierobežojumi, dažādi pieņēmumi un atkarības.
3. Funkcionālās prasības - detalizēti tiek aprakstītas minētās produkta funkcijas.

4. Nefunkcionālās prasības – tiek aprakstītas produkta prasības, kas attiecas uz kopējo sistēmas darbību (veiktspēja, uzturamība, drošība).

## 1.2. Vispārējs apraksts

### 1.2.1. Produkta perspektīva

"Sociālo mediju ziņojumu plūsmu pārskata vietne" ir veidota kā tīmekļa vietne, kas atradīsies iekš SAP HANA CLOUD Platformas (skat. Attēls 1.1). Sistēma var tikt izmantota uz ierīcēm kurām ir iespēja lietot tīmekļa pārlūkprogrammu, jo izstrādātā programma spēj pielāgoties mazākiem displejiem un attiecīgi attēlo informāciju pēc displeju izmēriem (skat. Attēls 2.3.) un (skat. Attēls 2.4.).




*Attēls 1.1*

Produkta procesu plūsmas piemērs:

- lietotājs pieslēdzas izstrādātajai tīmekļa vietnei, kura atrodas *SAP HANA Cloud* platformā un vēlās apskatīt visas mediju sienas.
- šis pieprasījums nonāk līdz API, kas izsauc MediaWall servisu no kura notiek kommunikācija ar *SAP HANA Cloud* datubāzi un dokumentu servisu.
- pēc datu iegūšanas no datubāzes tie tiek atgriezti lietotājam un attēloti tīmekļa vietnē, kur lietotājs var izvēlēties apskatīt mediju sienas individuāli.

### 1.2.2. Produkta funkcijas

Programmatūras produktam jārealizē šādas funkcijas:

- Datu glabāšanu.
- Datu iegūšana pēc tīmekļa servisu izsaukšanas.
- Kļūdu paziņojumi gadījumā, ja nekorekti ieejas dati tīmekļa servisiem.
- Lietotāju autentifikācija, kuru nodrošina HANA Cloud Paltformas Identity serviss.
- Mediju sienu izveidošanu, dzēšanu, labošanu un attēlošanu.
- Sīkziņu izgūšana no vietnes Twitter, pēc iepriekš nodefinētiem kritērijiem
- Sīkziņu attēlošana tīmekļa vietnē.
- Veco sīkziņu dzēšana pēc meklēšanas parametru labošanas vai arī mediju sienas dzēšanas.
- Mediju sienu bilžu uzglabāšana MongoDB datubāzē.
- Automātiska mediju sienu sīkziņu atjaunošana, kas tiek atkārtota noteiktā laika intervālā.
- Manuāla mediju sienu sīkziņu atjaunošana.

### 1.2.3. Sistēmas lietotāji un to raksturiezīmes

Sistēmai ir tikai divas lietotāju grupas:

- Viesis – var būt pilnīgi jebkurš cilvēks, kuram ir pieeja interneta parlūka programmai un ir atvēris šo tīmekļa vietni.
- Administrators – lietotājs, kuram pēc autorizācijas sistēmā ar administratora tiesībām būs iespēja lietot šīs tīmekļa vietnes funkcijas, kuras nav pieejamas viesim.

Administrators spēj:

- Izveidot jaunu mediju ziņojumu plūsmu sienu.
- Atjaunot eksistējošas mediju ziņojumu plūsmas sienas:
    - logo,

    - nosaukumu,
    - parametrus pēc kuriem meklē sīkziņas.
- Apskatīt pieejamās mediju ziņojumu plūsmu sienas kopumā un katru detalizēti.
- Dzēst mediju sienu.

Viesis spēj:

- Apskatīt kādas mediju sienas ir izveidotas.
- Apskatīt katru mediju sienu individuāli, kur ir redzams to:
    - iegūtās sīkziņas,
    - nosaukums,
    - logo.
- Apskatīt katru mediju sienas sīkziņu uznirstošā logā.

### 1.2.4. Vispārējie ierobežojumi

Sistēmas funkcijām, kas ļauj izveidot, labot un dzēst mediju ziņojumu plūsmu sienu jābūt pieejamām tikai autentificētam un autorizētam lietotājam.

Servera puses darbību nodrošina *Apache Tomcat* Serveris.

Saziņa starp klientu un serveri notiek ar tīmekļu servisiem.

Lai vieglāk būtu popularizēt šo vietni tiek noteikti šādi ierobežojumi:

- Klienta puses darbība tiek nodrošināta, izmantojot tīmekļa pārlūku un ierīcei (datoram, viedtālrunim, planšetdatoram) jābūt interneta pieslēgumam.
- Tīmekļa pārlūka versijas – Chrome 50.0, Firefox 46.0, Opera 37.0.

### 1.2.5. Pieņēmumi un atkarības

Tiek pieņemts, ka autentifikācija notiek izmantojot Hana Cloud Platformu, kur var uzstādīt administratora autorizāciju. Ja notiek datu dzēšanas tad ieraksti tiek pilnībā izdzēsti no datubāzes. Lai lietotu šo tīmekļa vietni tiek pieņemts, ka sistēmas

lietotājiem ir nepieciešamas pamatzināšanas darbam ar datoru un ar interneta pārlūka programmmu. Tehnisku ierobežojumu dēļ tiek attēlotas tikai jaunākās 50 sīkziņas iekš mediju sienas, jo pēc ilgāka laika mediju siena var saturēt tūkstošiem sīkziņu.Sistēmas normālai darbībai tiek pieņemts, ka lietotāja gala iekārtas izmantotā interneta pārlūkprogramma atbalsta HTML5, CSS un tai ir iespējots JavaScript.

## 1.3. Funkcionālās prasības

Funkcionālās prasības tiek iedalītas divās daļās – klienta puses un servera puses prasības. Klienta puses daļā tiek aprakstītas klienta funkcijas. Tās tiek aprakstītas lietotāju stāstu veidā, jo ir vispārīgas un klientam vieglāk saprotamas. Servera puses daļā tiek aprakstītas servera funkcijas. Tās tiek aprakstītas tabulu veidā, lai varētu detalizētāk izprast produkta funkcionalitāti.

### 1.3.1. Klienta puses funkcionālās prasības

**Lietotāju stāsta šablons**

| | |
|---|---|
| Identifikators | Lietotāja stāsta identifikators |
| Stāsts | Lietotāja stāsts – apraksta nepieciešamo funkcionalitāti no attiecīgā sistēmas lietotāja skata punkta.<br>“Es, kā < loma >, vēlos < vēlme/mērķis >.” |
| Akceptēšanas kritēriji | Definēti kritēriji, pēc kuru izpildīšanās var uzskatīt, ka lietotāja stāsts ir pilnībā implementēts. |

#### 1.3.1.1. Apskatīt visas pieejamās mediju sienas

| | |
|---|---|
| Identifikators | USER_STORY_SEE_ALL_WALLS |
| Stāsts | Es, kā viesis, vēlos sākuma lapā redzēt visas pieejamās mediju sienas, lai uzreiz spētu apskatīt vai tīmekļa vietne piedāvā man interesējošu tematu par kuru vēlos iegūt informāciju. |
| Akceptēšanas kritēriji | Sākuma lapā tiek attēlotas mediju sienas ar nosaukumiem un to atsauces tagiem, lai vieglāk būtu izprast kāds saturs ir katrai šai mediju sienai.<br>Sākuma lapā var nonākt vai nu klikšķinot uz navigācijas pogas "Walls" vai arī tikko atverot tīmekļa vietni. |

#### 1.3.1.2. Apskatīt vienu mediju sienu.

| | |
|---|---|
| Identifikators | USERY_STORY_SEE_ONE_WALL |
| Stāsts | Es, kā viesis, vēlos apskatīt katru mediju sienu individuāli detalizētāk, lai iegūtu vairāk informācijas par izvēlēto tematu. |
| Akceptēšanas kritēriji | Mediju sienas individuālajā skatā tiek attēlota informācija ar mediju sienas nosaukumu, logo un tās sīkziņām. Katrai sīkziņai ir redzams tās:<br>● autors,<br>● autora bilde,<br>● teksts,<br>● favorītu skaits,<br>● retvītu skaits, |

| | |
|---|---|
| | un ja ir tad arī ir redzama pievienotā bilde pie sīkziņas.<br>Individuālo skatu iespējams apskatīt klikšķinot uz vēlamās mediju sienas no sākuma lapas skata vai arī noklikšķinot uz “Walls” un tad uz sev vēlamo mediju sienu. |

#### 1.3.1.3. Apskatīt vienu sīkziņu.

| | |
|---|---|
| Identifikators | USER_STORY_SEE_POST |
| Stāsts | Es, kā viesis, mediju sienas skatā, vēlos apskatīt katru sīkziņu individuāli. |
| Akceptēšanas kritēriji | Individuālās mediju sienas skatā tiek attēlotas sīkziņas, kuras tiek atlasītas pēc padotajiem parametriem sienas izveidošanas brīdī.<br>Lietotājam ir iespēja katru sīkziņu apskatīt atsevišķi, lai vieglāk būtu uztvert informāciju un redzēt mazliet vairāk nekā redzams visu sīkziņu skatā.<br>Sīkziņu apskatīt individuāli ir iespējams no izvēlētās mediju sienas skata. Noklikšķinot uz vēlamās sīkziņas izveidosies uznirstošais logs ar šo sīkziņu. |

### 1.3.2. Server puses funkcionālās prasības

#### 1.3.2.1. Pievienot jaunu mediju sienu:

Funkcijas mērķis: Sistēmas administrators var izveidot jaunu mediju sienu, kurā pēc padotajiem parametriem tiks pievienotas cilvēku sīkziņas un šo mediju sienu varēs apskatīt visi sistēmas lietotāji. Administratoram ir iespēja meklēt sīkziņas pēc šiem trīs tipiem:

- Hashtag – atsauces tags, kas atrodas sīkziņās.
- incWord – vārdi kuriem jāatrodas atrastajās sīkziņās.
- exclWord – vārdi kuri nedrīkst atrasties iegūtajās sīkziņās.

| **Piekļuves tiesības:** |
|---|
| Administratoram |
| **Identifikators:** |
| MEDIAWALL_CREATE |
| **Ievade:** |
| • **wallname** – Mediju sienas nosaukums (teksts (32), unikāls, obligāts lauks)<br>• **logo** – Mediju sienas attēls (teksts (255), nav obligāts lauks)<br>• **searchInfo** – Masīvs ar sīkziņu meklēšanas kritērijiem:<br>  ○ **site** – Informācijas meklēšanas avots: Twitter, Instagram (teksts (255), obligāts lauks)<br>  ○ **type** – Informācijas meklēšanas tips: Hashtag, incWord, exclWord (teksts (255), obligāts lauks)<br>  ○ **value** – Informācijas meklēšanas parametru lauks (teksts (255), obligāts lauks) |

| |
|---|
| **Apstrāde:** |
| 1. Mediju sienas dati tiek pārbaudīti:<br>1.1. Vai obligātie lauki ir aizpildīti.<br>1.2. Vai ievaddati atbilst datu tipam.<br>1.3. Vai nepārsniedz maksimālo garumu.<br>2. Pēc pārbaudēm dati tiek saglabāti. |
| **Izvade:** |
| • Paziņojums par mediju sienas izveidošanu. |
| **Kļūdas paziņojumi:** |
| • Paziņojums, ka nav aizpildīts obligātais lauks.<br>• Paziņojums, ka ievaddati ir kļūdaini aizpildīti.<br>• Paziņojums, ka lauks pārsniedz maksimālo garumu.<br>• Mediju siena ar šadu sienas nosaukumu jau eksistē. |

#### 1.3.2.2. Dzēst mediju sienu.

Funkcijas mērķis: Sistēmas administrators spēj dzēst eksistējošās mediju sienas.

| |
|---|
| **Piekļuves tiesības:** |
| Administratoram |
| **Identifikators:** |
| MEDIAWALL_DELETE |
| **Ievade:** |

| |
|---|
| • **wallId** – Mediju sienas identifikators (vesels skaitlis, obligāts lauks) |
| **Apstrāde:** |
| • Tiek dzēsta izvēlētā mediju siena, tai attiecīgās sīkziņas un sienas logo. |
| **Izvade:** |
| • Mediju siena tika veiksmīgi izdzēsta. |
| **Kļūdas paziņojumi:** |
| • Mediju siena ar šādu identifikatoru neeksistē. |

#### 1.3.2.3. Atjaunot mediju sienu.

Funkcijas mērķis: Sistēmas administrators spēj atjaunot izvēlētās mediju sienas datus ar jaunajiem viņa padotajiem datiem.

| |
|---|
| **Piekļuves tiesības:** |
| Administratoram |
| **Identifikators:** |
| MEDIAWALL_UPDATE |
| **Ievade:** |
| • **wallname** – Mediju sienas nosaukums (teksts (32), unikāls, obligāts lauks)<br>• **logo** – Mediju sienas attēls (teksts (255), nav obligāts lauks)<br>• **searchInfo** – Masīvs ar sīkziņu meklēšanas kritērijiem:<br>  ○ **site** – Informācijas meklēšanas avots: Twitter, Instagram (teksts (255), obligāts lauks)<br>  ○ **type** – Informācijas meklēšanas tips: Hashtag, incWord, exclWord (teksts |

| |
|---|
| (255), obligāts lauks)<br>○ **value** – Informācijas meklēšanas parametru lauks (teksts (255), obligāts lauks) |
| **Apstrāde:** |
| ● Mediju sienas dati tiek pārbaudīti:<br>○ Vai obligātie lauki ir aizpildīti.<br>○ Vai ievaddati atbilst datu tipam.<br>○ Vai nepārsniedz maksimālo garumu.<br>● Pēc pārbaudēm dati tiek saglabāti. |
| **Izvade:** |
| • Paziņojums par mediju sienas labošanu. |
| **Kļūdas paziņojumi:** |
| • Paziņojums, ka nav aizpildīts obligātais lauks.<br>• Paziņojums, ka ievaddati ir kļūdaini aizpildīti.<br>• Paziņojums, ka lauks pārsniedz maksimālo garumu.<br>• Mediju siena ar šadu sienas nosaukumu jau eksistē. |

#### 1.3.2.4. Atjaunot visu mediju sienu sīkziņas.

Funkcijas mērķis: Sistēmas administrators spēj atjaunot mediju sienas iegūstot jaunas sīkziņas.

| |
|---|
| **Piekļuves tiesības:** |
| Administratoram. |

| **Identifikators:** |
| --- |
| MEDIAWALL_UPDATE_POSTS |
| **Ievade:** |
| • Šai funkcijai ievaddatu nav. |
| **Apstrāde:** |
| • Visu eksistējošo mediju sienu sīkziņas tiek atjaunotas ar jaunākajām sīkziņām. |
| **Izvade:** |
| • Tiek atgriezts, ka mediju sienu sīkziņas atjaunot izdevās veiksmīgi. |
| **Kļūdas paziņojumi:** |
| • Radās problēmas ar sīkziņu atjaunošanu. |

#### 1.3.2.5. Atjaunot vienas mediju sienas logo.

Funkcijas mērķis: Sistēmas administrators spēj pievienot un dzēst izvēlētās mediju sienas logo.

| **Piekļuves tiesības:** |
| --- |
| Administratoram. |
| **Identifikators:** |
| MEDIAWALL_UPDATE_LOGO |
| **Ievade:** |
| • **wallId** – Mediju sienas identifikators (vesels skaitlis, obligāts lauks) |

| |
|---|
| • **logo** – Mediju sienas attēls (teksts (255), nav obligāts lauks) |
| **Apstrāde:** |
| ● Mediju sienas dati tiek pārbaudīti:<br>○ Vai šāda mediju siena eksistē.<br>○ Vai logo jau eksistē priekš izvēlētās mediju sienas.<br>■ Jā: Iepriekšējais logo tiek dzēsts un tiek pievienots jaunais logo.<br>■ Nē: Tiek pievienots jaunais logo.<br>● Pēc pārbaudēm dati tiek saglabāti. |
| **Izvade:** |
| • Paziņojums, ka izdevās atjaunot logo. |
| **Kļūdas paziņojumi:** |
| • Paziņojums, ka ievaddati ir kļūdaini aizpildīti.<br>• Mediju siena ar šadu ID neeksistē. |

#### 1.3.2.6. Apskatīt visas pieejamās mediju sienas.

Funkcijas mērķis: Visiem sistēmas lietotājiem dot iespēju apskatīt visas eksistējošās mediju sienas.

| |
|---|
| **Piekļuves tiesības:** |
| Administratoram, viesim. |
| **Identifikators:** |
| MEDIAWALL_WALLS |

| **Ievade:** |
| --- |
| • Šai funkcijai ievaddatu nav. |
| **Apstrāde:** |
| ● No datubāzes tiek atlasītas visas izveidotās mediju sienas. |
| **Izvade:** |
| ● Tiek atgriezti mediju sienas dati:<br>○ Nosaukums.<br>○ Atsauces tags (*Hashtag*). |

#### 1.3.2.7. Atrast mediju sienu pēc identifikatora.

Funkcijas mērķis: Visiem sistēmas lietotājiem dot iespēju apskatīt vienu mediju sienu detalizētāk, izsaucot MEDIAWALL_WALLS funkciju un noklišķinot uz vēlamo sienu.

| **Piekļuves tiesības:** |
| --- |
| Administratoram, viesim. |
| **Identifikators:** |
| MEDIAWALL_WALL_ID |
| **Ievade:** |
| • **wallId** – Mediju sienas identifikators (vesels skaitlis, obligāts lauks) |
| **Apstrāde:** |

| |
|---|
| • Izvēlētās mediju sienas dati tiek attēloti vienā lapā. |
| **Izvade:** |
| • Tiek atgriezti mediju sienas dati:<br>  ○ Logo.<br>  ○ Nosaukums.<br>  ○ Iegūtās sīkziņas:<br>  ■ URL uz autora attēlu.<br>  ■ Autora lietotājvārds.<br>  ■ Autora parādāmais vārds.<br>  ■ Teksts.<br>  ■ Favorītu skaits.<br>  ■ Retvītu skaits. |

#### 1.3.2.8. Atrast mediju sienu pēc nosaukuma.

Funkcijas mērķis: Sistēmas administratoram dot iespēju apskatīt vienu mediju sienu detalizētāk, meklējot sienu pēc tās nosaukuma.

| |
|---|
| **Piekļuves tiesības:** |
| Administratoram. |
| **Identifikators:** |
| MEDIAWALL_WALL_NAME |
| **Ievade:** |
| • **wallname** – Mediju sienas nosaukums (teksts (32), unikāls, obligāts lauks) |
| **Apstrāde:** |

| |
|---|
| • Izvēlētās mediju sienas dati tiek attēloti vienā lapā. |
| **Izvade:** |
| • Tiek atgriezti mediju sienas dati:<br>  ○ Logo.<br>  ○ Nosaukums.<br>  ○ Iegūtās sīkziņas:<br>  ■ Autora konta attēls.<br>  ■ Autora lietotājvārds.<br>  ■ Autora parādāmais vārds.<br>  ■ Teksts.<br>  ■ Favorītu skaits.<br>  ■ Retvītu skaits. |

#### 1.3.2.9. Apskatīt vienas mediju sienas logo.

Funkcijas mērķis: Atgriezt logo attiecīgi pēc pieprasītās mediju sienas id.

| |
|---|
| **Piekļuves tiesības:** |
| Administratoram, viesim. |
| **Identifikators:** |
| MEDIAWALL_WALL_LOGO |
| **Ievade:** |
| • **wallId** – Mediju sienas identifikators (vesels skaitlis, obligāts lauks) |
| **Apstrāde:** |
| 1. Tiek meklēta izvēlētā mediju siena pēc id. |

| |
|---|
| 2. Pēc mediju sienas logo saglabātās simbolu virknes tiek meklēts tai attiecīgais logo. |
| **Izvade:** |
| • Tiek atgriezts mediju sienas logo. |
| **Kļūdas paziņojumi:** |
| • Mediju siena ar šadu identifikatoru neeksistē.<br>• Mediju sienai neeksistē logo. |

### 1.3.2.10. Dzēst vienas mediju sienas logo.

Funkcijas mērķis: Dzēst logo attiecīgi pēc pieprasītās mediju sienas id.

| |
|---|
| **Piekļuves tiesības:** |
| Administratoram |
| **Identifikators:** |
| MEDIAWALL_DELETE_LOGO |
| **Ievade:** |
| • **wallId** – Mediju sienas identifikators (vesels skaitlis, obligāts lauks) |
| **Apstrāde:** |
| • Tiek dzēsts logo izvēlētajai mediju sienai. |
| **Izvade:** |
| • Mediju sienas logo tika veiksmīgi izdzēsts. |

| **Kļūdas paziņojumi:** |
| --- |
| • Mediju siena ar šādu identifikatoru neeksistē. |

## 1.4. Nefunkcionālās prasības

### 1.4.1 Veikstpējas prasības

Tīmekļa vietnei jānodrošina vairāki paralēli lietotāji, kā arī jānodrošina vairāku mediju sienu uzturēšana.

### 1.4.2 Pieejamība

Sistēmai jābūt pieejamai un funkcionējošai visu laiku, tas ir 24 stundas diennaktī un 7 dienas nedēļā. Sistēmai var rasties izņēmuma gadījums, kas ir apkopes laikā, kad tā var būt nepieejama.

### 1.4.3. Ārējās saskarnes prasības

Tīmekļa vietnei jābūt viegli pārskatāmai un ērti lietojamai. Tīmekļa vietnes dizainam jābūt dinamiskam, lai uz mazākām ierīcēm būtu iespējams apskatīt informāciju tik pat efektīvi kā uz lielākām.

### 1.4.4. Drošības prasības

Par paroļu glabāšanu, administratoru autentifikāciju un izveidošanu atbild *SAP HANA Cloud Identity* serviss, kuru nodrošina *SAP HANA Cloud* platforma.

### 1.4.5. Citas prasības

Servera pusē tīmekļa servisi ir balsīti un izstrādāti ievērojot *RESTful* tīmekļa servisu arhitektūras vadlīnijas.

### 1.4.6. Uzturamības prasības

Jāveic rezervju kopiju izveide, lai programmkoda lielu kļūdu vai datu zuduma gadījumā būtu iespējams iegūt iepriekš izveidoto programmatūras produktu.

# 2. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

## 2.1. Ievads

### 2.1.1. Nolūks

Šajā dokumentā tiks parādīts programmatūras prasību specifikācijā noteikto funkcionālo un nefunkcionālo prasību projektējums.

### 2.1.2. Darbības sfēra

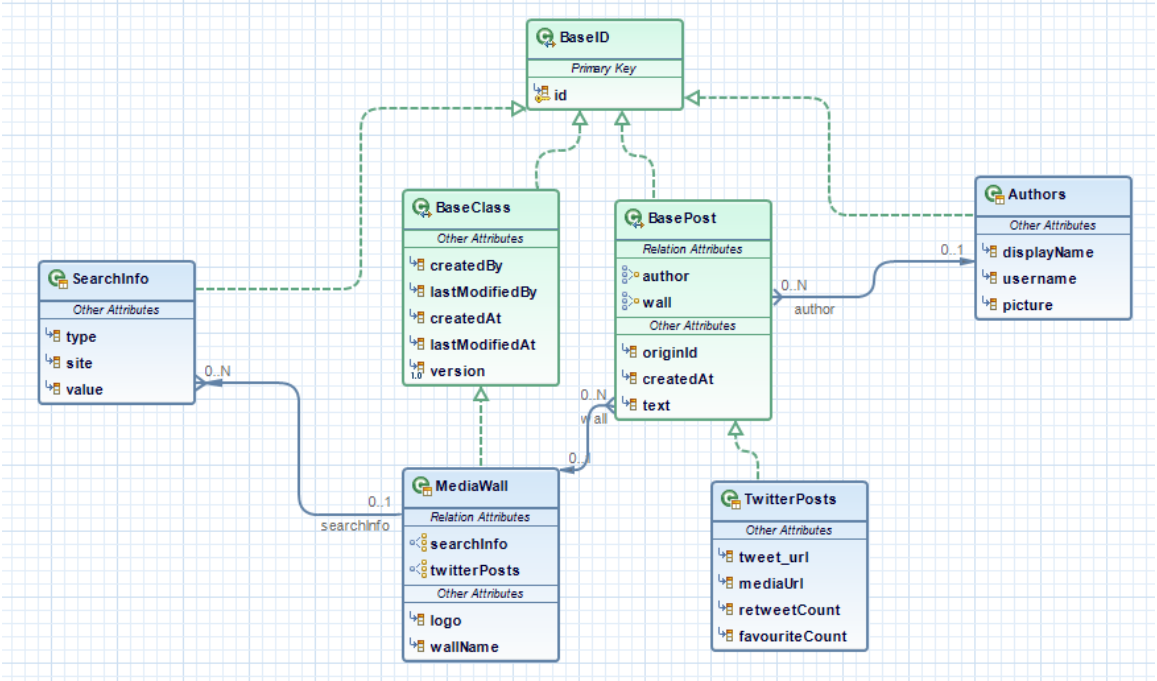
"Sociālo mediju ziņojumu plūsmu pārskata vietne" nodrošina dažādu mediju sienu izveidi, dzēšanu un atjaunošanu, kā arī pašu mediju sienu individuālu apskati. Tiek nodrošināta iespēja administratoram ievadīt vairāku veidu parametrus, lai precīzāk spētu iegūt informāciju vienā skatā, ko administrators ir vēlējies. Nodrošināta arī iespēja labot datubāzes informāciju pēc kuras notiek sīkziņu meklēšana.

## 2.2. Datu bāzes projektējums

### 2.2.1. Java datu modelis

Datu modelis tika izveidots iekš rīka eclipse. Java datu modelis satur trīs bāzes klases un četras atvasinātas klases. Bāzes klases tika veidotas ar nodomu, lai būtu iespējama atkārtota izmantojamība, jo programmatūras produktam ir iespējami vēl daudz uzlabojumu, piemēram, datu iegūšana no citas vietnes.

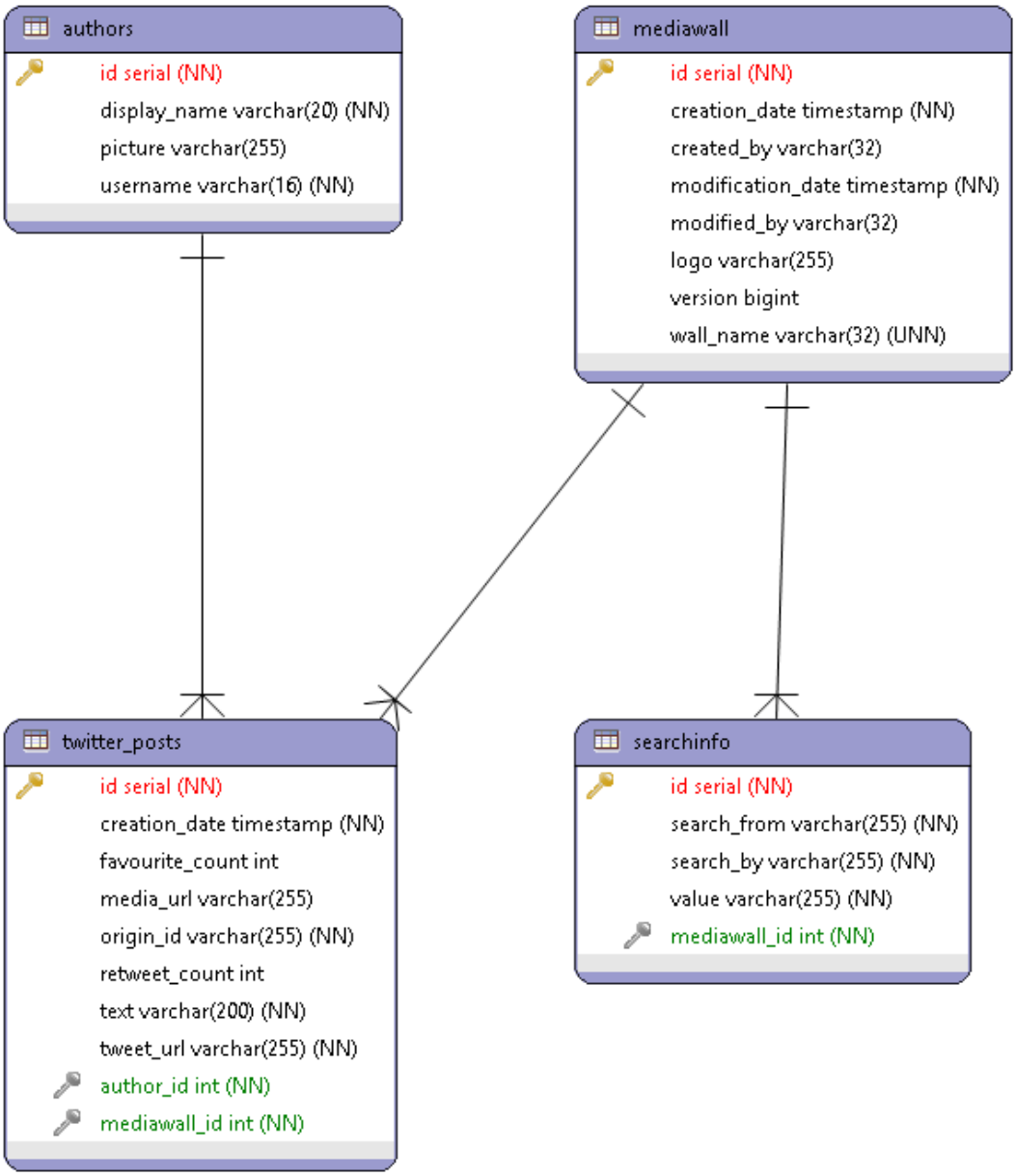
- Ar BaseID klasi tika paredzēts, ka katrai klasei nav obligāti jābūt identifikatoram.
- Ar BaseClass klasi tika paredzēts, ka katrai klasei nav obligāti jābut izveidošanas datumam un lietotājam kurš to izveidojis.
- Ar BasePost klasi tika paredzēts, lai veidojot jaunus avotus no kā meklēt informāciju, nenāktos lieki atkārtoties jaunizveidotajā klasē.




*2.1. att.*, **Java datu modelis**

### 2.2.2. Datu bāzes ER modelis.

Datu modelis tika ģenerēts ar JPA rīkiem, kas datubāzē izskatās kā redzams attēlā (skat. Attēls 2.2.). Salīdzinot ar java datu modeli ir redzams, ka mediawall tabulai ir visi BaseClass lauki un tās nav divas tabulas, kas arī attīecīgi ir redzams ar pārējām tabulām.




*2.2. att.*, **Fiziskais modelis**

*NN - obligāts lauks, *UNN - unikāls lauks,  - primārās atslēgas lauks,  - ārējās atslēgas lauks.

### 2.2.3. Datubāzes tabulas

Sistēmas dati tiek glabāti 4 tabulās, kuras glabājās vienā datubāzē. Tabulas satur informāciju par mediju sienu, tās sīkziņu meklēšanas parametriem, iegūtajām sīkziņām un sīkziņu autoriem. Viens specifiskais lauks ir mediju sienas logo, jo tajā glabājas simbolu virkne pēc kuras attiecīgi meklē pašu bildi, kas ir saglabāta iekš mongoDB ar dokumentu servisa palīdzību. Otrs specifiskais lauks ir mediju sienas versija, jo tā automātiski mainās pēc administratora veiktajiem atjauninājumiem un tas tika izveidots ar iemeslu, lai vairāki administratori vienlaicīgi nevarētu pārrakstīt cits cita izmaiņas veicot labojumus.

Izmantotie apzīmējumi tabulās:

- PK – (primārā atslēga) - unikāls lauks datubāzē.
- FK – (atsauces atslēga) - lauks, kurš norāda uz kādas citas tabulas primāro atslēgu.
- Iespējamie datu tipi:
    - int – datu tips veselu skaitļu glabāšanai.
    - timestamp – datu tips datumu glabāšanai.
    - varchar – datu tips simbolu glabāšanai
    - bigint – datu tips lielu skaitļu glabāšanai.

#### 2.2.3.1. Tabula “mediawall”

Tabula „mediawall” satur datus par izveidotajām sociālo mediju sienām.

| Nr.p.k | Lauka nosaukums | Datu tips | Obligāts | Atslēga | Apraksts |
|---|---|---|---|---|---|
| 1. | id | int | Jā | PK | Mediju sienas unikāls identifikators, automātiski ģenerēts. |
| 2. | creation_date | timestamp | Jā | - | Mediju sienas izveidošanas datums, automātiski ģenerēts. |
| 3. | created_by | varchar(32) | Jā | - | Mediju sienas izveidotājs. |
| 4. | modification_date | timestamp | Jā | - | Mediju sienas labojuma datums, automātiski ģenerēts. |
| 5. | modified_by | varchar(32) | Jā | - | Mediju sienas labotājs. |
| 6. | logo | varchar(255) | Nē | - | Mediju sienas attēls. |
| 7. | version | bigint | Jā | - | Mediju sienas versija, automātiski ģenerēts. |
| 8. | wall_name | varchar(32) | Jā | - | Mediju sienas unikāls nosaukums. |

#### 2.2.3.2. Tabula “twitter_posts”

Tabula „twitter_posts” satur datus par Twitter sīkziņu, kurš ir tās autors un pie kuras mediju sienas tā atrodas.

| Nr.p.k | Lauka nosaukums | Datu tips | Obligāts | Atslēga | Apraksts |
|---|---|---|---|---|---|
| 1. | id | int | Jā | PK | Twitter ziņas unikāls identifikators, automātiski ģenerēts. |
| 2. | creation_date | timestamp | Jā | - | Twitter ziņas izveidošanas datums. |
| 3. | favourite_count | int | Jā | - | Twitter ziņas favorītu skaits(?) |
| 4. | media_url | varchar(255) | Nē | - | Twitter ziņas bildes tīmekļa adrese. |
| 5. | origin_id | varchar(255) | Jā | - | Twitter tvīta unikāls identifikators. |
| 6. | retweet_count | int | Jā | - | Twitter ziņas retvītu skaits . |
| 7. | text | varchar(200) | Jā | - | Twitter ziņas teksts. |
| 8. | tweet_url | varchar(255) | Jā | - | Twitter ziņas tīmekļa adrese . |
| 9. | authors_id | int | Jā | FK | Ārējā atslēga uz “authors” tabulu. |
| 10. | mediawall_id | int | Jā | FK | Ārējā atslēga uz “mediawall” tabulu. |

#### 2.2.3.3. Tabula “authors”

Tabula „authors” satur detalizētus datus par Twitter sīkziņas autoru.

| Nr.p.k | Lauka nosaukums | Datu tips | Obligāts | Atslēga | Apraksts |
|---|---|---|---|---|---|
| 1. | id | int | Jā | PK | Autora unikāls identifikators, automātiski ģenerēts. |
| 2. | display_name | varchar(20) | Jā | - | Autora parādāmais vārds. |
| 3. | picture | varchar(255) | Jā | - | Autora konta attēls. |
| 4. | username | varchar(16) | Jā | - | Autora lietotājvārds. |

#### 2.2.3.4. Tabula “searchinfo”

Tabula „searchinfo” satur datus pēc kuriem notiek Twitter sīkziņu meklēšana un kurai mediju sienai pieder noteiktie meklēšanas parametri.

| Nr.p.k | Lauka nosaukums | Datu tips | Obligāts | Atslēga | Apraksts |
|---|---|---|---|---|---|
| 1. | id | int | Jā | PK | Informācijas meklēšanas unikāls identifikators, automātiski ģenerēts. |
| 2. | search_from | varchar(255) | Jā | - | Informācijas meklēšanas avots: twitter, instagram, facebook. |

| | | | | | |
|---|---|---|---|---|---|
| 3. | search_by | varchar(255) | Jā | - | Informācijas meklēšanas kritērijs. |
| 4. | value | varchar(255) | Jā | - | Informācijas meklēšanas parametri. |
| 5. | mediawall_id | int | Jā | FK | Ārējā atslēga uz "mediawall" tabulu. |

## 2.3. Tīmekļu servisu realizācija

| **Funckijas ID:** | |
|---|---|
| MEDIAWALL_CREATE | |
| **Pieprasījums:** | |
| • Vienotais resursu vietrādis (URL) -**/MediaWall/rest/walls/**<br>• HTTP pieprasījums – **POST**<br>• Ievades formats – **Application/JSON** | |
| **Padotie parametri:** | **Ievade:** |

| | |
|---|---|
| **wall** – padoti dati JSON datu struktūras formātā pēc kuriem tiks izveidota mediju siena. Dati JSON formātā atrodas POST ķermeni (POST body) pieprasījuma brīdī. | <pre>{<br>  "searchInfo": [<br>  {<br>  "site": "TWITTER",<br>  "type": "HASHTAG",<br>  "value": "food"<br>  },<br>  {<br>  "site": "TWITTER",<br>  "type": "INCWORD",<br>  "value": "delicious"<br>  },<br>  {<br>  "site": "TWITTER",<br>  "type": "EXCLWORD",<br>  "value": "bad"<br>  }<br>  ],<br>  "wallName": "Food wall"<br>}</pre> |

| **Atbilde:** | |
|---|---|
| **Iespējamie rezultāti:** | **Izvade:** |
| Atgriež HTTP statusu ar kodu "201 Created” - mediju siena tika veiksmīgi izveidota. | Atgriež izveidotās mediju sienas datus JSON formātā. |
| Atgriež HTTP statusu ar kodu “400 bad request” – neizdevās izveidot mediju sienu. | - |

| **Funckijas ID:** |
|---|

| MEDIAWALL_DELETE | |
|---|---|
| **Pieprasījums:** | |
| • Vienotais resursu vietrādis (URL) -**/MediaWall/rest/walls/{id}**<br>• HTTP pieprasījums – **DELETE** | |
| **Padotie parametri:** | **Ievade:** |
| **id –** identifikators, kas tiek padots URL kā query parameters, ar kura palīdzību noteiks kura mediju siena tiks dzēsta. | Piemēram:<br>**/MediaWall/rest/walls/1** |
| **Atbilde:** | |
| **Iespējamie rezultāti:** | **Izvade:** |
| Atgriež HTTP statusu ar kodu "202 OK” - mediju siena tika veiksmīgi izdzēsta. | - |
| Atgriež HTTP statusu ar kodu “404 NOT FOUND” – neizdevās izdzēst mediju sienu. | - |

| **Funckijas ID:** | |
|---|---|
| MEDIAWALL_DELETE_LOGO | |
| **Pieprasījums:** | |
| • Vienotais resursu vietrādis (URL) -**/MediaWall/rest/walls/{id}/logo**<br>• HTTP pieprasījums – **DELETE** | |
| **Padotie parametri:** | **Ievade:** |

| | |
|---|---|
| **id –** identifikators, kas tiek padots URL kā query parameters, ar kura palīdzību noteiks kura mediju siena tiks dzēsta. | Piemēram:<br>**/MediaWall/rest/walls/5/logo** |
| **Atbilde:** | |
| **Iespējamie rezultāti:** | **Izvade:** |
| Atgriež HTTP statusu ar kodu "200 OK" - mediju sienas logo tika veiksmīgi izdzēsts. | - |
| Atgriež HTTP statusu ar kodu "404 NOT FOUND" – mediju siena ar tādu id neeksistē. | - |

| | |
|---|---|
| **Funckijas ID:** | |
| MEDIAWALL_UPDATE | |
| **Pieprasījums:** | |
| • Vienotais resursu vietrādis (URL) -**/MediaWall/rest/walls/{id}**<br>• HTTP pieprasījums – **PUT**<br>• Ievades formats – **Application/JSON** | |
| **Padotie parametri:** | **Ievade:** |
| **id –** identifikators, kas tiek padots URL kā query parameters, ar kura palīdzību noteiks kura mediju siena jāatjauno. | {<br>"searchInfo": [<br>{<br>"site": "TWITTER",<br>"type": "HASHTAG", |

| | |
|---|---|
| **wall** – padoti dati JSON datu struktūras formātā pēc kuriem tiks izveidota mediju siena. Dati JSON formātā atrodas POST ķermeni (POST body) pieprasījuma brīdī. | <pre>"value": "foods"<br>}<br>],<br>"wallName": "Food wall"<br>}</pre> |

| **Atbilde:** | |
|---|---|
| **Iespējamie rezultāti:** | **Izvade:** |
| Atgriež HTTP statusu ar kodu "200 OK" - mediju sienas dati tika veiksmīgi atjaunoti. | Atgriež mediju sienas datus JSON formātā. |
| Atgriež HTTP statusu ar kodu "404 NOT FOUND" – mediju siena ar tādu id neeksistē. | |

| **Funckijas ID:** |
|---|
| MEDIAWALL_UPDATE_POSTS |
| **Pieprasījums:** |
| • Vienotais resursu vietrādis (URL) -**/MediaWall/rest/walls/update**<br>• HTTP pieprasījums – **PUT** |
| **Atbilde:** |

| **Iespējamie rezultāti:** | **Izvade:** |
|---|---|
| Atgriež HTTP statusu ar kodu "200 OK" - mediju sienas dati tika veiksmīgi atjaunoti. | Atgriež mediju sienas datus JSON formātā. |
| Atgriež HTTP statusu ar kodu "404 NOT FOUND" – neatrada nevienu mediju sienu, lai spētu atjaunot datus . | |

| **Funckijas ID:** | |
|---|---|
| MEDIAWALL_UPDATE_LOGO | |
| **Pieprasījums:** | |
| • Vienotais resursu vietrādis (URL) **-/MediaWall/rest/walls/{id}/logo**<br>• HTTP pieprasījums – **PUT**<br>• Ievades formats – **Multipart/form-data** | |
| **Padotie parametri:** | **Ievade:** |
| **id –** identifikators, kas tiek padots URL kā query parameters, ar kura palīdzību noteiks kuras mediju sienas logo jāatjauno.<br><br>**logo –** mediju sienas jaunā bilde. Jaunā bilde PUT pieprasījumā jāatrodas form-data daļā. | **Test.png** – attēla formāta fails. |
| **Atbilde:** | |
| **Iespējamie rezultāti:** | **Izvade:** |
| Atgriež HTTP statusu ar kodu "200 OK” - mediju sienas logo tika veiksmīgi atjaunots. | Atgriež mediju sienas datus JSON formātā. |
| Atgriež HTTP statusu ar kodu “404 NOT FOUND” – mediju siena ar tādu id neeksistē. | - |

| **Funckijas ID:** |
|---|

| MEDIAWALL_WALLS | |
|---|---|
| **Pieprasījums:** | |
| • Vienotais resursu vietrādis (URL) -**/MediaWall/rest/walls**<br>• HTTP pieprasījums – **GET** | |
| **Atbilde:** | |
| **Iespējamie rezultāti:** | **Izvade:** |
| Atgriež HTTP statusu ar kodu "200 OK" – Mediju sienu dati tika veiksmīgi atgriezti. | Atgriež visu mediju sienu datus JSON formātā. |
| Atgriež HTTP statusu ar kodu "404 NOT FOUND" – netika atrasta neviena mediju siena. | - |

| **Funckijas ID:** | |
|---|---|
| MEDIAWALL_WALL_ID | |
| **Pieprasījums:** | |
| • Vienotais resursu vietrādis (URL) -**/MediaWall/rest/walls/{id}**<br>• HTTP pieprasījums – **GET** | |
| **Padotie parametri:** | **Ievade:** |
| **id –** identifikators, kas tiek padots URL kā query parameters, ar kura palīdzību noteiks kuras mediju sienas logo jāatgriež. | Piemēram:<br>**/MediaWall/rest/walls/1** |
| **Atbilde:** | |

| **Iespējamie rezultāti:** | **Izvade:** |
|---|---|
| Atgriež HTTP statusu ar kodu "200 OK” - mediju sienas logo tika veiksmīgi atjaunots. | Atgriež mediju sienas datus JSON formātā. |
| Atgriež HTTP statusu ar kodu “404 NOT FOUND” – mediju siena ar tādu id neeksistē. | |

| **Funckijas ID:** | |
|---|---|
| MEDIAWALL_WALL_NAME | |
| **Pieprasījums:** | |
| • Vienotais resursu vietrādis (URL) -**/MediaWall/rest/walls/name/{name}**<br>• HTTP pieprasījums – **GET** | |
| **Padotie parametri:** | **Ievade:** |
| **name –** sienas nosaukums, kas tiek padots URL kā query parameters, ar kura palīdzību noteiks kura mediju siena jāatgriež. | Piemēram:<br>**/MediaWall/rest/walls/name/iPhones** |
| **Atbilde:** | |

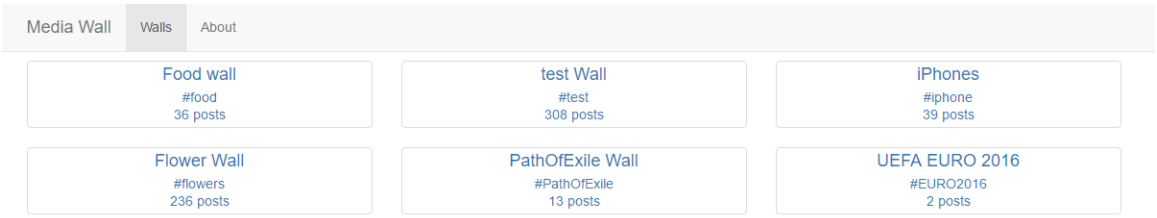
| **Iespējamie rezultāti:** | **Izvade:** |
|---|---|
| Atgriež HTTP statusu ar kodu "200 OK” - mediju sienas dati tika veiksmīgi atgriezti. | Atgriež mediju sienas datus JSON formātā. |
| Atgriež HTTP statusu ar kodu “404 NOT FOUND” – mediju siena ar tādu nosaukumu neeksistē. | |

| **Funckijas ID:** | |
|---|---|
| MEDIAWALL_WALL_LOGO | |
| **Pieprasījums:** | |
| • Vienotais resursu vietrādis (URL) -**/MediaWall/rest/walls/{id}/logo**<br>• HTTP pieprasījums – **GET** | |
| **Padotie parametri:** | **Ievade:** |
| **id –** identifikators, kas tiek padots URL kā query parameters, ar kura palīdzību noteiks kuras mediju sienas logo jāatgriež. | Piemēram:<br>**/MediaWall/rest/walls/1/logo** |
| **Atbilde:** | |
| **Iespējamie rezultāti:** | **Izvade:** |
| Atgriež HTTP statusu ar kodu "200 OK" - mediju sienas logo tika veiksmīgi atgriezts. | Atgriež attēla formāta failu |
| Atgriež HTTP statusu ar kodu "404 NOT FOUND" – mediju sienai ar tādu id neeksistē logo. | - |

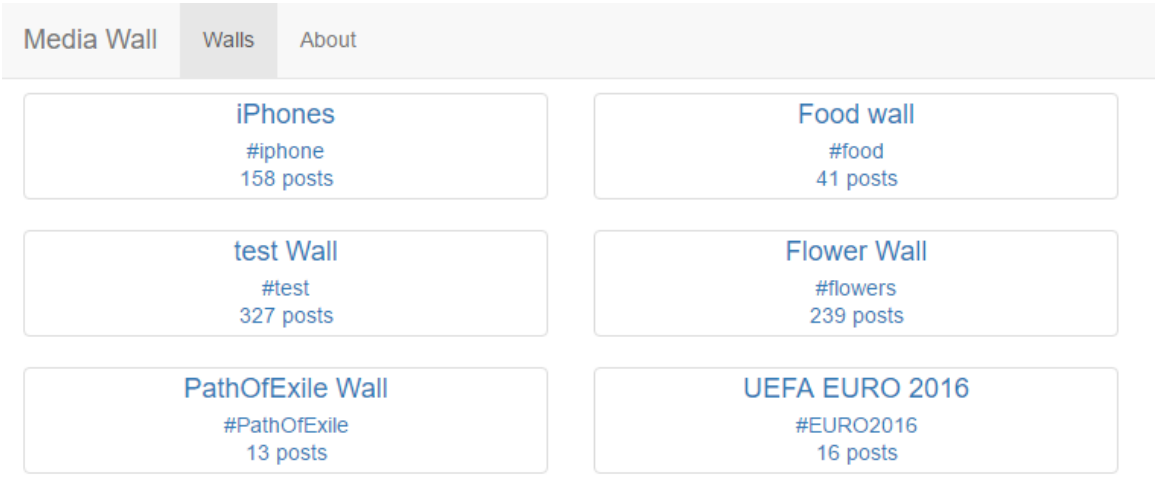
## 2.4. Ārējā saskarne

### 2.4.1. Viesu saskarne

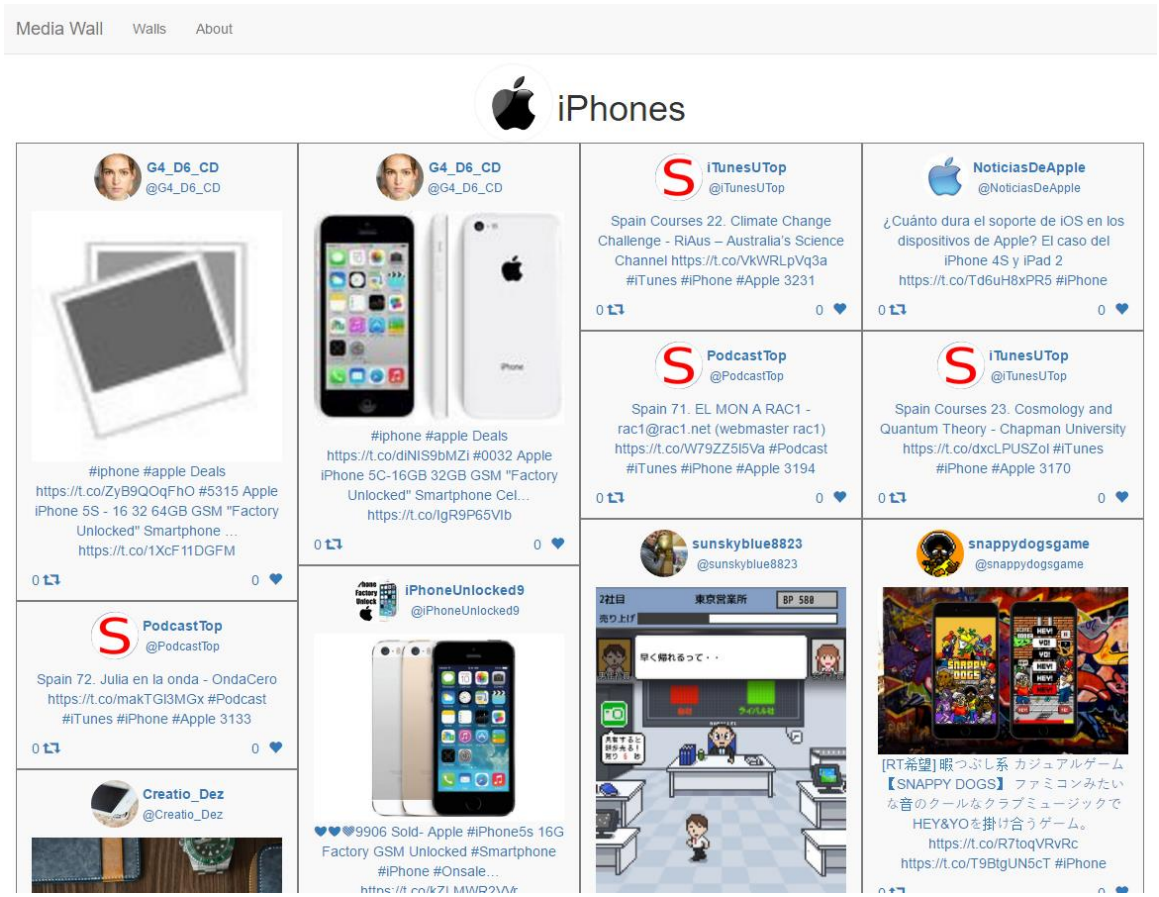
Lietotājs ar sistēmu saskarsies caur pārlūkprogrammu. Lietotāja saskarne būs mājas lapa ar navigāciju, kur lietotājs spēs apskatīt pieejamās sienas kopumā, kā arī katru atsevišķi detalizētāk ar tās sīkziņām. Lai attēlotu reaģējošu tīmekļa lapas dizainu tika ievietoti vairāki attēli uz dažādiem pārlūkprogrammas izmēriem.



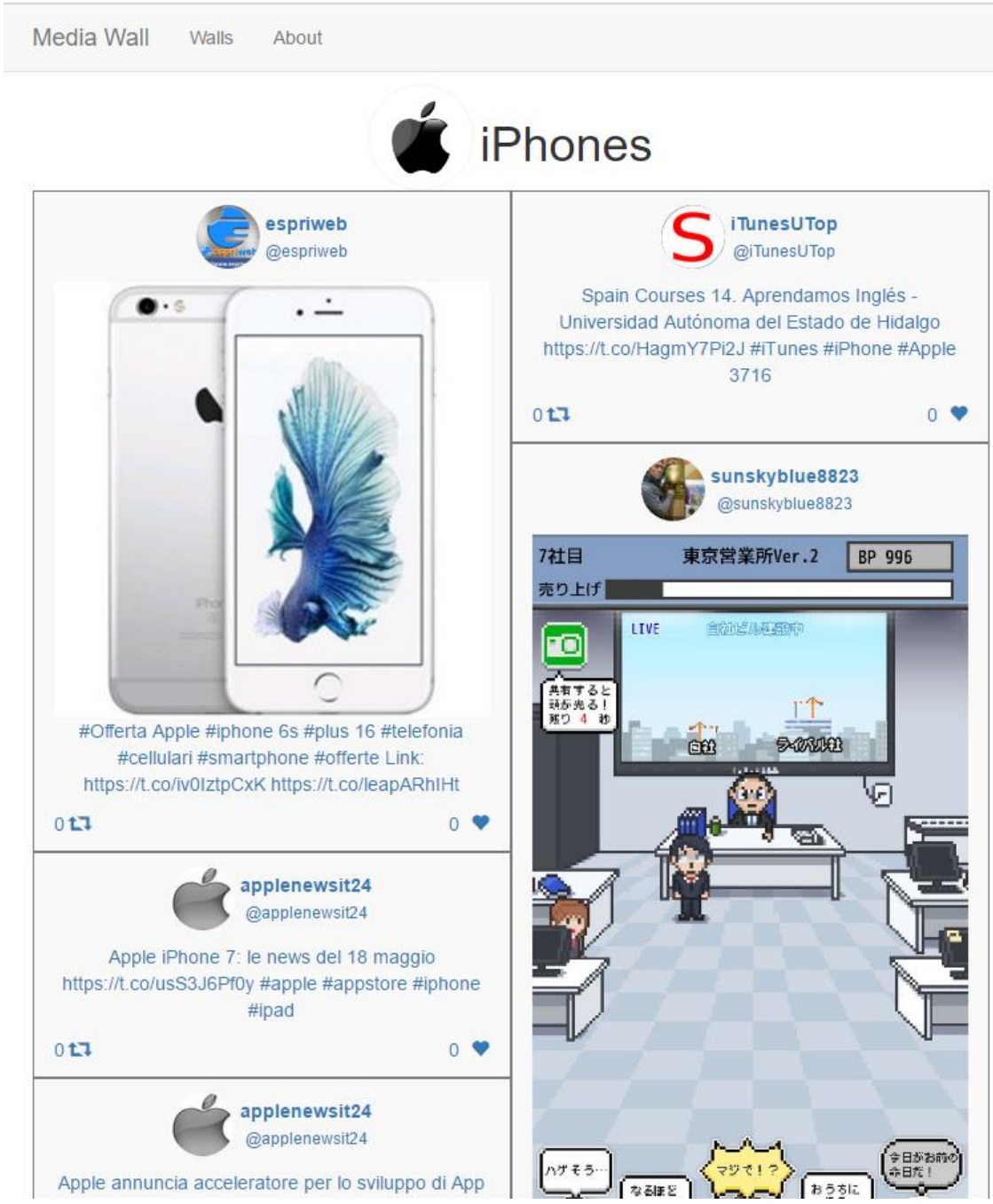

*2.3. att.,* **Mediju sienu saraksta skats pilnā ekrānā**



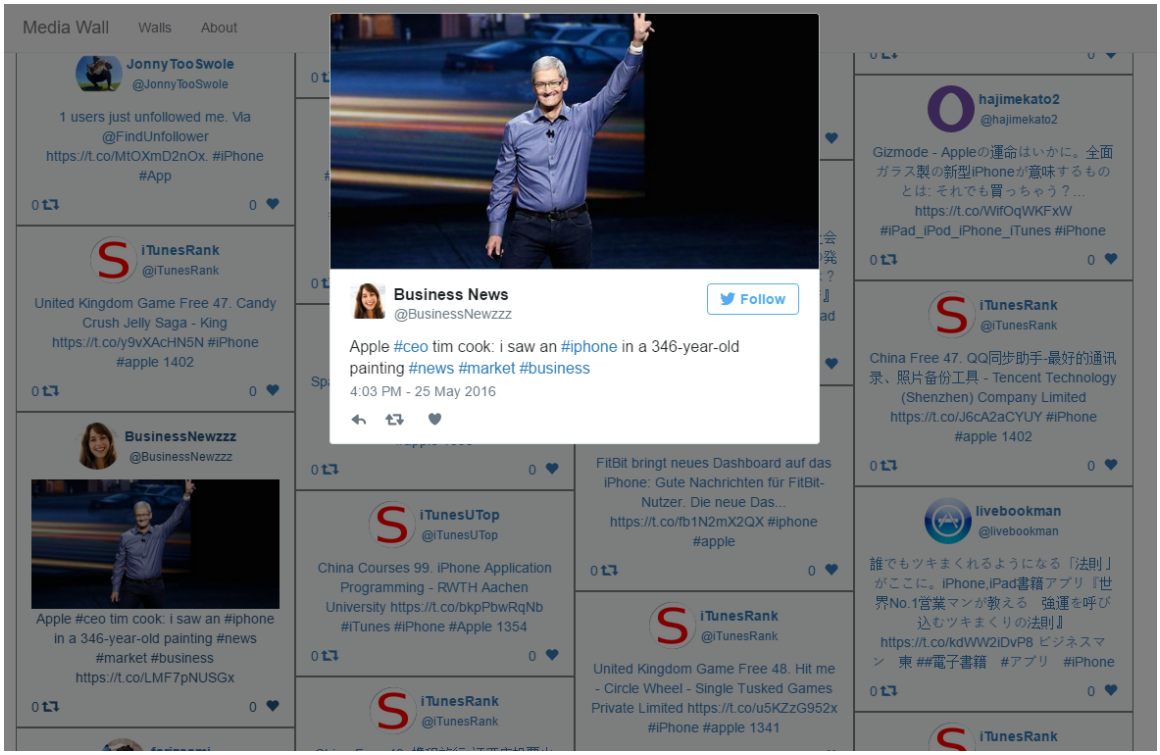

*2.4. att.,* **Mediju sienu saraksta skats**



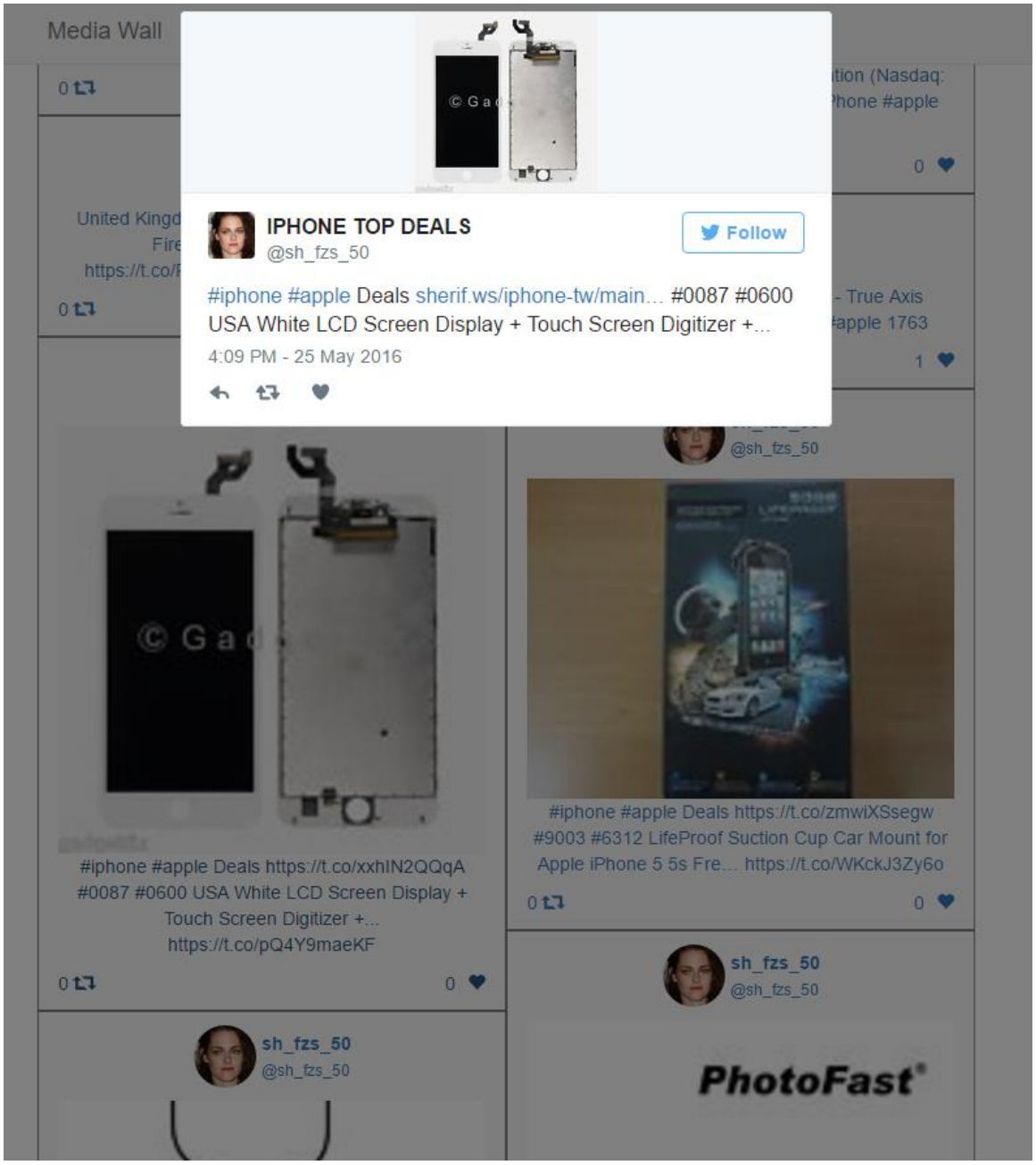

*2.5. att.,* **Vienas mediju sienas skats.**




*2.6. att.*, **Vienas mediju sienas.**




*2.6. att.*, **Vienas mediju sienas sīkziņas skats.**




*2.7. att.,* **Vienas mediju sienas sīkziņas skats.**

### 2.4.2. Administratora saskarne

Administratoram tika paredzēts, ka ar sistēmu saskarsies caur pārlūkprogrammu, kur būs iespēja doties uz administratora paneli. Administratora paneļa saskarnē būtu formas, navigācija un saraksti, taču mājaslapa, kurā tiks attēlota informācija izskatīsies tā pat kā viesim. Administratora paneli neizdevās izstrādāt paredzētājā laikā, tāpēc produkta funkcijas, kuras spēj izmantot administrators jāveic

caur paplašinājumu, kas sniedz iespēju veikt *POST, GET, PUT* un *DELETE* pieprasījumus.

# 3. TESTĒŠANAS DOKUMENTĀCIJA

## 3.1. Ievads

Testēšanas dokumentācijas nolūks ir fiksēt testēšanas gaitu un testu rezultātus. Testēšanas procesā tika izmantoti automātiskie vienībtesti un manuālie testi, kuri atviegloja pirmkoda uzlabošanu projekta izstrādes laikā. Ar automātisko testu palīdzību tika pārbaudīta galvenā programmas funkcionalitāte, kas ļauj izveidot, dzēst un atjaunot mediju sienu datus datubāzē. Pirms dati nonāca līdz datubāzei tika arī pārbaudīta datu validācija, kura nodrošina korektu datu ievadi.

Ar manuālo testu palīdzību tika pārbaudīta mediju sienu bilžu rediģēšana, dzēšana un attēlošana, lai vizuāli varētu pārliecināties, ka tiešām veiktās izmaiņas ir veiksmīgi izdevušās un paredzētā funkcionalitāte strādā.

Pirms katra projekta būvējuma veikšānas tiek izmantoti JUnit testi, nodrošinot pārbaudi galvenajai programmatūras funkcionalitātei.

## 3.2. Galvenie testēšanas kritēriji

- Testēšanai jāizmanto cita datu bāze nekā produkcijas versijai, lai neietekmētu produkcijas datus,
- Jābūt automatizētiem vienībtestiem,

## 3.3. Veiktie testpiemēri

### 3.3.1. Automatizētie vienībtesti

#### 3.3.1.1. Mediju sienas izveide.

| **Funkcijas ID:** | | | |
|---|---|---|---|
| MEDIAWALL_CREATE | | | |
| Nr. | Darbību apraksts | Sagaidāmais rezultāts | Rezultāts |
| 1. | Izveido mediju sienu aizpildot visus laukus un saglabā to datubāzē. | Mediju sienas izveide notika veiksmīgi un datubāzē tika atrasta izveidotā mediju siena. | OK. 23.05.2016 |
| 2. | Mediju sienas izveide ar sienas nosaukumu, kur garums pārsniedz 32 simbolus. | Paziņo, ka mediju sienas nosaukumam jābūt garākam par 1 simbolu un īsākam par 32 simboliem. | OK. 23.05.2016 |
| 3. | Mediju sienas izveide ar sienas nosaukumu, kur garums ir 0 simbolu. | Paziņo, ka mediju sienas nosaukumam jābūt garākam par 1 simbolu un īsākam par 32 simboliem. | OK. 23.05.2016 |
| 4. | Mediju sienas izveide neaizpildot meklēšanas parametrus pēc kuriem meklē sīkziņas. | Iegūts kļūdas paziņojumu, ka meklēšanas parametri nedrīkst būt tukši. | OK. 23.05.2016 |
| 5. | Mediju sienas izveide aizpildot tikai vienu meklēšanas parametru un salabā to datubāzē. | Mediju sienas izveide notika veiksmīgi un datubāzē tika atrasta izveidotā mediju siena. | OK. 23.05.2016 |
| 6. | Mediju sienas izveide neaizpildot | Paziņo, ka mediju sienas nosaukums | OK. |

| | | | |
|---|---|---|---|
| | sienas nosaukumu. | nevar būt tukš. | 23.05.2016 |

### 3.3.1.2. Mediju sienas dzēšana.

| **Funkcijas ID:** | | | |
|---|---|---|---|
| MEDIAWALL_DELETE | | | |
| Nr. | Darbību apraksts | Sagaidāmais rezultāts | Rezultāts |
| 1. | Izveidota mediju siena, kurai pirms un pēc tās dzēšanas pārbauda vai tā vēl atrodas datubāzē. | Pirms mediju sienas dzēšanas izveidotā siena tika atrasta un pēc dzēšanas netika atrasta. | OK. 23.05.2016 |

### 3.3.1.3. Mediju sienas labošana.

| **Funkcijas ID:** | | | |
|---|---|---|---|
| MEDIAWALL_UPDATE | | | |
| Nr. | Darbību apraksts | Sagaidāmais rezultāts | Rezultāts |
| 1. | Mediju sienas nosaukumu labošana uz garumu kas pārsniedz 32 simbolus. | Paziņo, ka mediju sienas nosaukumam jābūt garākam par 1 simbolu un īsākam par 32 simboliem. | OK. 23.05.2016 |
| 2. | Mediju sienas nosaukumu labošana uz garumu 0 simboli. | Paziņo, ka mediju sienas nosaukumam jābūt garākam par 1 simbolu un īsākam par 32 | OK. 23.05.2016 |

| | | | |
|---|---|---|---|
| | | simboliem. | |
| 3. | Mediju sienas labošana izdzēšot nosaukumu. | Paziņo, ka mediju sienas nosaukums nevar būt tukš. | OK. 23.05.2016 |
| 4. | Mediju sienas labošana izdzēšot meklēšanas parametrus. | Paziņo ka meklēšanas parametri nedrīkst būt tukši. | OK. 23.05.2016 |
| 5. | Mediju sienas labošana izmainot meklēšanas parametrus un atjaunojot sienu datubāzē. | Izdevās izveidot jaunus meklēšanas parametrus izvēlētai mediju sienai. | OK. 23.05.2016 |

#### 3.3.1.4. Mediju sienas sīkziņu labošana.

| **Funkcijas ID:** | | | |
|---|---|---|---|
| MEDIAWALL_UPDATE_POSTS | | | |
| Nr. | Darbību apraksts | Sagaidāmais rezultāts | Rezultāts |
| 1. | Izveido mediju sienu.<br>Palaiž funkciju, kas iegūst mediju sienu sīkziņas. Pārbauda vai ir iegūtas sīkziņas izveidotajai sienai. | Mediju sienai veiksmīgi pievienotas sīkziņas. | OK. 23.05.2016 |

#### 3.3.1.5. Visu mediju sienu apskatīšana.

| **Funkcijas ID:** | | | |
|---|---|---|---|
| MEDIAWALL_WALLS | | | |
| Nr. | Darbību apraksts | Sagaidāmais rezultāts | Rezultāts |
| 1. | Pēc divu Mediju sienu izveidošanas tiek pārbaudīts vai izsaucot visas sienas atgriež | Tiek atgrieztas divas mediju sienas. | OK. 23.05.2016 |

| | | | |
|---|---|---|---|
| | divas. | | |

#### 3.3.1.6. Mediju sienas apskatīšana pēc id.

| **Funkcijas ID:** | | | |
|---|---|---|---|
| MEDIAWALL_WALL_ID | | | |
| Nr. | Darbību apraksts | Sagaidāmais rezultāts | Rezultāts |
| 1. | Izveido mediju sienu ar id (99) un saglabā to datubāzē. Pēctam pārbauda vai meklējot sienu ar id 99 atgriež tikko izveidoto sienu. | Nav kļūdu paziņojumu un ir izdevies atrast sienu pēc identifikatora. | OK. 23.05.2016 |

#### 3.3.1.7. Mediju sienas apskatīšana pēc nosaukuma.

| **Funkcijas ID:** | | | |
|---|---|---|---|
| MEDIAWALL_WALL_NAME | | | |
| Nr. | Darbību apraksts | Sagaidāmais rezultāts | Rezultāts |
| 1. | Izveido mediju sienu ar nosaukumu “Test” un saglabā to datubāzē. Pēc tam pārbauda vai meklējot sienu pēc nosaukuma “Test” atrod tikko izveidoto sienu. | Nav kļūdu paziņojumu un ir izdevies atrast sienu pēc tās nosaukuma. | OK. 23.05.2016 |

### 3.3.2. Manuālie testi

#### 3.3.2.1. Mediju sienas logo labošana.

| **Funkcijas ID:** | | | |
|---|---|---|---|
| MEDIAWALL_UPDATE_LOGO | | | |
| Nr. | Darbību apraksts | Sagaidāmais rezultāts | Rezultāts |
| 1. | Izvēlās mediju sienu pēc identifikatora, kurai vēlās atjaunot logo. Padod bildi, kuru vēlās kā jauno logo. | Iepriekšējais mediju sienas logo tiek aizstāts ar jauno mediju sienas logo. | OK. 23.05.2016 |

#### 3.3.2.2. Mediju sienas logo dzēšana.

| **Funkcijas ID:** | | | |
|---|---|---|---|
| MEDIAWALL_DELETE_LOGO | | | |
| Nr. | Darbību apraksts | Sagaidāmais rezultāts | Rezultāts |
| 1. | Izvēlās mediju sienu pēc identifikatora, kurai vēlās dzēst logo. | Eksistējošais logo tiek dzēsts un mediju sienai tiek pievienots noklusējuma logo. | OK. 23.05.2016 |

#### 3.3.2.3. Mediju sienas logo apskatīšana.

| **Funkcijas ID:** | | | |
|---|---|---|---|
| MEDIAWALL_WALL_LOGO | | | |
| Nr. | Darbību apraksts | Sagaidāmais rezultāts | Rezultāts |
| 1. | Izvēlās mediju sienu, kurai ir pievienots logo. | Tiek attēlots izvēlētās mediju sienas logo. | OK. 23.05.2016 |
| 2. | Izvēlās mediju sienu, | Tiek atgriezta pēc noklusējuma pievienotā | OK. |

| | | | |
|---|---|---|---|
| | kurai neeksistē logo. | bilde, jo pieprasītai mediju sienai neeksistē logo. | 23.05.2016 |

# 4. PROJEKTA ORGANIZĀCIJA

Projekts tika izstrādāts pēc Agile izstrādes metodoloģijas. Izstrāde sākās ar to, ka galvenās prasības tika definētas vispārīgi, taču balstoties uz tām varēja sākt programmatūras izstrādi un laika gaitā konsultējoties ar klientu, programmatūras prasības tika pilnveidotas. Tā kā projektu izstrādāja viens cilvēks, tad arī nācās apgūt dažādus projekta izstrādes rīkus - *Maven*, *Sap Hana Cloud* platformu, *Spring* satvaru, *Twitter* programmsaskarni un *Backbone JS*. Protams projekta izstrādē noritēja arī ikdienas pārrunas ar darba vadītāju, kurās tika pārrunāti uzdevumi, kas ir izdarīti un kuri būtu jāizdara tuvāko dienu laikā. Uzdevumi, kas ir jādara arī bija pieejami problēmu un projektu sekošanas rīkā *JIRA*.

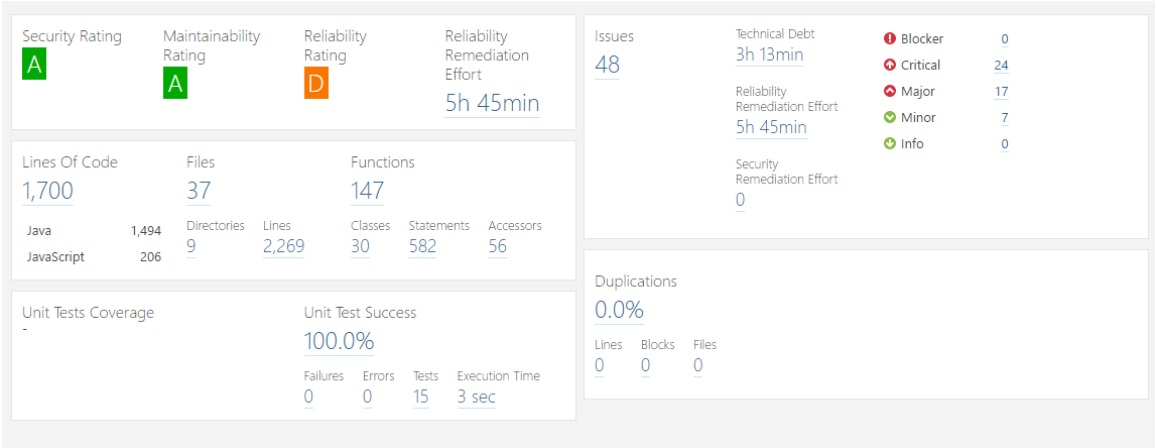
Izstrādātās programmatūras produkta dati sākotnēji tika glabāti relāciju datubāzē *PostgreSQL*, jo sākotnēji nebija zināmas visas prasības, bet radoties prasībai, ka vajadzētu glabāt bildes, tika pielietota vēl viena datubāze – *MongoDB*.

Viss programmatūras pirmkods tika rakstīts iekš izstrādes vides *Eclipse*. Ar Eclipse paplašinājumu palīdzību tika veiktas pirmkoda versiju uzglabāšanas iekš *SVN* repozitorijas.

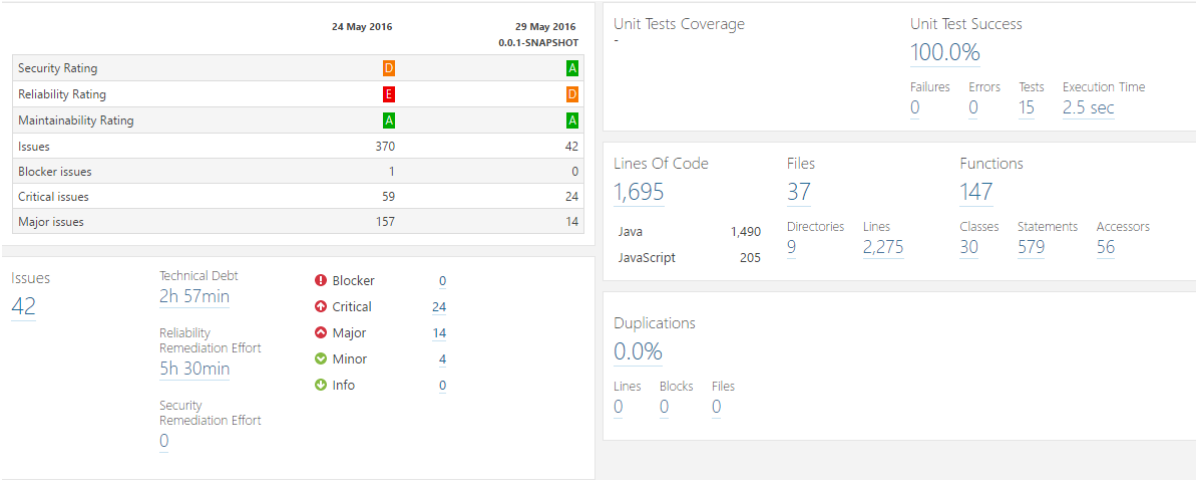
# 5. KVALITĀTES NODROŠINĀŠANA

Lai izstrādātajam programmas produktam nodrošinātu pēc iespējas labāku sistēmas kvalitāti, tika veiktas sekojošas darbības:

- iepazīšanās ar *REST* arhitektūras vadlīnijām,
- regulāras pārrunas ar prakses vadītāju,
- programmkoda validācija izmantojot *SonarQube* (skat. Attēls 5.1.) un (skat. Attēls 5.2.),
- veikta pirmkoda versiju glabāšana izmantojot *Subversion*,
- automātisko testu izveide,
- manuālā testēšana.



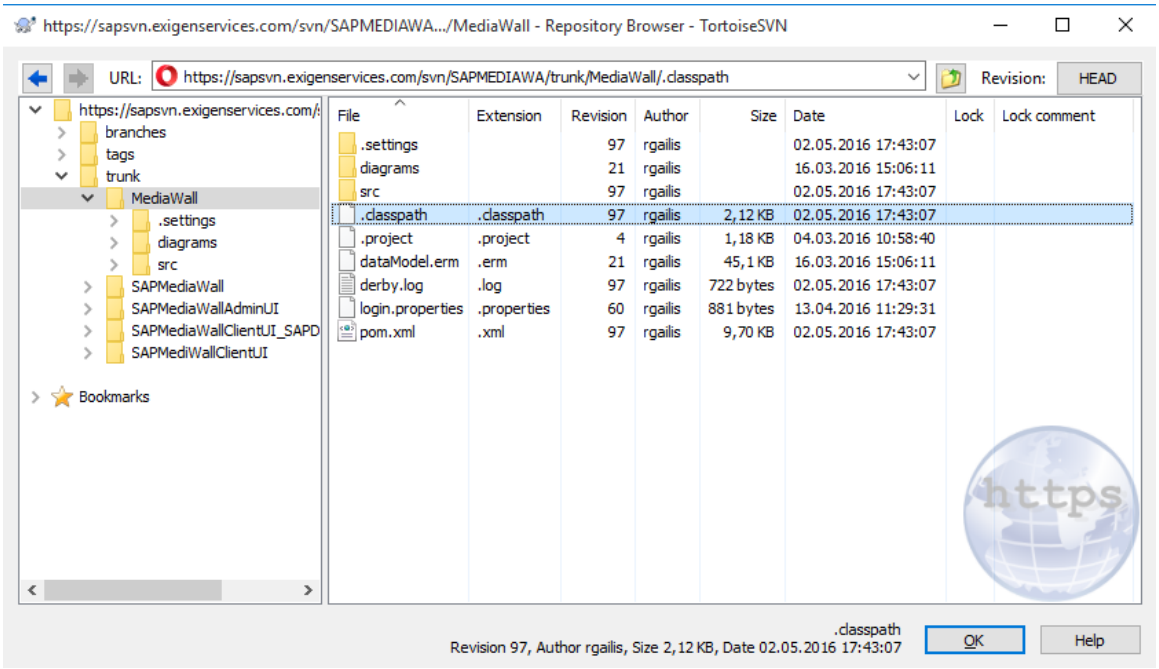

*5.1. att.,* **Iegūtie rezultāti no *SonarQube*.**




*5.2. att.,* **Iegūtie rezultāti no *SonarQube* pēc labojumiem.**

# 6. KONFIGURĀCIJU PĀRVALDĪBA

Projektam tika izmantots versiju pārvaldības rīks Subversion (SVN), kurā tika glabātas dažādas pirmkoda versijas. Sistēmas pirmkods repozitorijā tika pievienots pēc katras nozīmīgākās izmaiņas projektā. Lai versiju kontrole būtu pārskatāmāka, tad pie katras pirmkoda versijas pievienošanas tika pievienoti JIRA numuri.




*6.1. att.,* **Programmas pirmkoda uzglabāšanas repozitorijs.**

| Revision | Date | Changes | Author |
|---|---|---|---|
| 97 | 02/05/2016, 17:43 | 10 | rgailis |
| 93 | 27/04/2016, 11:23 | 25 | rgailis |
| 60 | 13/04/2016, 11:29 | 72 | rgailis |
| ***36** | **24/03/2016, 14:58** | **15** | **rgailis** |
| 35 | 22/03/2016, 15:19 | 8 | rgailis |
| 34 | 22/03/2016, 11:23 | 19 | rgailis |
| 22 | 16/03/2016, 15:45 | 2 | rgailis |
| 21 | 16/03/2016, 15:06 | 33 | rgailis |
| 4 | 04/03/2016, 10:58 | 89 | rgailis |
| 3 | 04/03/2016, 10:58 | 1 | rgailis |

*6.2. att.,* **Programmas pirmkoda uzglabāšanas vēsture.**

# 7. DARBIETILPĪBAS NOVĒRTĒJUMS

Pirms katras produkta daļas izstrādes tika veikts aptuvens novērtējums cik laika tas varētu aizņemt un pēc tam salīdzināts ar cik laika tas aizņēma patiesībā, atsaucoties uz ierakstiem, kas tika veikti katru nedēļu iekš uzņēmuma darba laika uzskaites tabulas.

| Darbība | Plānotais laiks | Reālais laiks | Starpība |
|---|---|---|---|
| **PPS izveide** | 40 stundas | 40 stundas | 0 |
| **PPA izveide** | 40 stundas | 40 stundas | 0 |
| **Testēšanas dokumentācija** | 30 stundas | 40 stundas | -10 stundas |
| **Servisu izveide** | 60 stundas | 70 stundas | -10 stundas |
| **Modeļu izveide** | 40 stundas | 40 stundas | 0 |
| **Datu piekļuves objektu izveide** | 50 stundas | 60 stundas | -10 stundas |
| **Validācija** | 60 stundas | 80 stundas | -20 stundas |
| **Automātiskie testi** | 110 stundas | 90 stundas | +20 stundas |
| **Lietotāju saskarne** | 90 stundas | 100 stundas | -10 stundas |
| **Kopā:** | 520 stundas | 560 stundas | -40 stundas |

Darbietilpības plānotais laiks ar reāli patērēto laiku atšķirās šādu faktoru dēļ:

- nācās apgūt jaunas tehnoloģijas,
- negaidītu kļūdu rašanās,
- maza iepriekšējā pieredze projektu plānošanā un izstrādē.

# 8. PROGRAMMATŪRAS KODA PIEMĒRI

## 8.1. Programmatūras API

```
1. /**
2. * The MediaWalls program implements an application that
3. * simply allows you to create Media walls that are filled with
   people's tweets.
4. * Each wall is displayed on separate page that allows people to
    read specific information
5. * with less effort.
6. *
7. * @author  Rihards Gailis
8. * @version 1.0
9. * @since   22.02.2016
10.      */
11.      @Component
12.      @Path("/walls")
13.      @Produces({ MediaType.APPLICATION_JSON })
14.      public class MediaWalls {
15.
16.          /**
17.          * Autowiring fields allows us not to worry about the
18.          * bean creation in spring configuration file.
19.          */
20.          @Autowired
21.          MediaWallService mediaWallSrv = null;
22.
23.          @Autowired
24.          TwitterService twitterSrv = null;
25.
26.          private static final Logger logger = LoggerFactory.ge
   tLogger(MediaWalls.class);
27.
28.          /**
29.          * This method is used to get all existing Media walls
   .
30.          *
31.          * @return Success: This returns list of all existing
   walls and (Status code: OK).
32.          *         Failure: Media Wall was not found (Status c
   ode: NOT_FOUND).
33.          */
34.          @GET
35.          public Response getWalls() {
36.              Response retVal;
```

```
37.             List<MediaWall> walls = mediaWallSrv.getAllWalls(
    );
38.             if (walls != null)
39.                 retVal = Response.ok(walls).status(Status.OK)
    .build();
40.             else
41.                 retVal = Response.status(Status.NOT_FOUND).bu
    ild();
42.             return retVal;
43.         }
44.
45.         /**
46.         * This method is used to create Media Wall without lo
    go.
47.         * It takes passed parameter wall in json format.
48.         *
49.         * @param wall This is MediaWall object which contains

50.         *             Wall name, SearcInfo such as hashtags,
    words to include and exclude.
51.         *
52.         * @return MediaWall This returns created Media Wall a
    nd (Status code: CREATED).
53.         */
54.         @POST
55.         @Consumes(MediaType.APPLICATION_JSON)
56.         public Response createWall(@Valid MediaWall wall) {
57.             wall = mediaWallSrv.createWall(wall);
58.
59.             InputStream logo = getClass().getResourceAsStream
    ("/images/default.jpg");
60.             wall.setLogo(mediaWallSrv.saveWallImage(wall.getW
    allName(), logo));
61.             wall = mediaWallSrv.updateWall(wall);
62.
63.             twitterSrv.updateWallTweets(wall);
64.             return Response.ok(wall).status(Status.CREATED).b
    uild();
65.         }
66.
67.         /**
68.         * This method is used to get Media Wall by passing an
     existing ID.
69.         * It takes passed parameter id from url and searches
    if Media wall with asked id exists.
70.         *
71.         * @param id This is MediaWall's ID.
72.         *
73.         * @return Success: This returns Media Wall object and
     (Status code: OK).
74.         *         Failure: This returns (Status code: NOT_FOU
    ND).
```

```
75.          */
76.          @GET
77.          @Path("/{id}")
78.          public Response getWall(@PathParam("id") Integer id)
   {
79.              Response retVal;
80.
81.              MediaWall wall = mediaWallSrv.getWallById(id);
82.              if (wall == null) {
83.                  retVal = Response.status(Status.NOT_FOUND).bu
   ild();
84.              } else {
85.                  retVal = Response.ok(wall).build();
86.              }
87.
88.              return retVal;
89.          }
90.
91.          /**
92.          * This method is used to update Media Wall.
93.          * It takes passed parameters id and wall to know whic
   h wall is going to be updated
94.          * and which fields to update.
95.          *
96.          * @param id This is MediaWall's ID.
97.          * @param wall This is MediaWall object which contains

98.          *             Wall name, SearcInfo such as hashtags,
   words to include and exclude.
99.          * @return Success: This returns updated Media Wall an
   d (Status code: OK).
100.         *         Failure: This returns (Status code: NOT_FOU
   ND).
101.         */
102.         @PUT
103.         @Path("/{id}")
104.         @Consumes(MediaType.APPLICATION_JSON)
105.         public Response updateWall(@PathParam("id") Integer i
   d, @Valid MediaWall wall) {
106.             Response retVal;
107.
108.             MediaWall existingWall = mediaWallSrv.getWallById
   (id);
109.
110.             if (wall == null) {
111.                 retVal = Response.status(Status.NOT_FOUND).bu
   ild();
112.             } else {
113.                 existingWall.mergeWall(wall);
114.                 mediaWallSrv.updateWall(existingWall);
115.                 retVal = Response.ok(existingWall).build();
116.             }
```

```
117.
118.             return retVal;
119.         }
120.
121.         /**
122.         * This method is used to delete single Media Wall.
123.         * It takes passed parameter id to know which wall is
   going to be deleted.
124.         *
125.         * @param id This is MediaWall's ID.
126.         * @return Success: This returns (Status code: OK).
127.         *         Failure: This returns (Status code: NOT_FOU
   ND).
128.         */
129.         @DELETE
130.         @Path("/{id}")
131.         public Response deleteWall(@PathParam("id") Integer i
   d) {
132.             Response retVal;
133.
134.             MediaWall wall = mediaWallSrv.getWallById(id);
135.             if (wall != null) {
136.                 mediaWallSrv.deleteWall(wall);
137.                 retVal = Response.status(Status.OK).build();

138.             } else {
139.                 retVal = Response.status(Status.NOT_FOUND).bu
   ild();
140.             }
141.             return retVal;
142.         }
143.
144.         /**
145.         * This method is used to update all Media Walls with
   new posts.
146.         *
147.         * @return Success: This returns (Status code: OK).
148.         *         Failure: This returns (Status code: NOT_FOU
   ND).
149.         */
150.         @PUT
151.         @Path("/update")
152.         public Response updateWalls() {
153.             Response retVal;
154.
155.             if (mediaWallSrv.getAllWalls() != null) {
156.                 twitterSrv.updateAllWallTweets();
157.                 retVal = Response.status(Status.OK).build();

158.             } else {
159.                 retVal = Response.status(Status.NOT_FOUND).bu
   ild();
```

```
160.             }
161.
162.             return retVal;
163.         }
164.
165.         /**
166.         * This method is used to get Media Wall by passing an
    existing Wall name.
167.         * It takes passed parameter name from url and searche
   s if Media wall with asked Wall name exists.
168.         *
169.         * @param name This is MediaWall's name.
170.         *
171.         * @return Success: This returns Media Wall object and
    (Status code: OK).
172.         *         Failure: This returns (Status code: NOT_FOU
   ND).
173.         */
174.         @GET
175.         @Path("/name/{name}")
176.         public Response getWall(@PathParam("name") String nam
   e) {
177.             Response retVal;
178.
179.             MediaWall wall = mediaWallSrv.getWallByName(name)
   ;
180.             if (wall == null) {
181.                 retVal = Response.status(Status.NOT_FOUND).bu
   ild();
182.             } else {
183.                 retVal = Response.ok(wall).status(Status.OK).
   build();
184.             }
185.
186.             return retVal;
187.         }
188.
189.         /**
190.         * This method is used to get Media Wall's logo by pas
   sing an existing ID.
191.         * It takes passed parameter id from url and searches
   if Media wall with asked id exists.
192.         *
193.         * @param id This is MediaWall's ID.
194.         *
195.         * @return Success: This returns logo and (Status code
   : OK).
196.         *         Failure: This returns (Status code: NOT_FOU
   ND).
197.         */
198.         @GET
199.         @Path("/{id}/logo")
```

```
200.         public Response showWallLogo(@PathParam("id") Integer
    id) {
201.             Response retVal;
202.
203.             MediaWall wall = mediaWallSrv.getWallById(id);
204.
205.             if (wall != null) {
206.                 String wallName = wall.getWallName();
207.                 if (wall.getLogo() == null || wall.getLogo().
   isEmpty()) {
208.                     retVal = Response.status(Status.NOT_FOUND
   ).build();
209.                 } else {
210.                     retVal = Response.ok(DocumentService.getD
   ocumentAsByteArray(wallName)).status(Status.OK)
211.                                 .type(mediaWallSrv.getWallImageTy
   pe(wallName)).build();
212.                 }
213.             } else {
214.                 retVal = Response.status(Status.NOT_FOUND).bu
   ild();
215.             }
216.             return retVal;
217.         }
218.
219.         /**
220.         * This method is used to update Media Wall's logo by
   passing an existing ID.
221.         * It takes passed parameter id from url and searches
   if Media wall with asked id exists.
222.         * If Wall with such id exists then it takes passed im
   age and sets it for specified wall.
223.         *
224.         * @param id This is MediaWall's ID.
225.         * @param logo This is Image file which will be displa
   yed alongside wall's name.
226.         * @return Success: This returns Media Wall with its u
   pdated information and (Status code: OK).
227.         *         Failure: This returns (Status code: NOT_FOU
   ND).
228.         */
229.         @PUT
230.         @Consumes(MediaType.MULTIPART_FORM_DATA)
231.         @Path("/{id}/logo")
232.         public Response updateLogoForWall(@Multipart(value =
   "logo", type = "image/*") InputStream logo,
233.                 @PathParam("id") Integer id) {
234.             Response retVal;
235.
236.             MediaWall wall = mediaWallSrv.getWallById(id);
237.             if (wall == null) {
```

```
238.                 retVal = Response.status(Status.NOT_FOUND).bu
   ild();
239.             } else {
240.                 if (wall.getLogo() == null || wall.getLogo().
   isEmpty()) {
241.                     logger.info("logo = null");
242.                     wall.setLogo(mediaWallSrv.saveWallImage(w
   all.getWallName(), logo));
243.                 } else {
244.                     logger.info("logo = not null");
245.                     mediaWallSrv.deleteWallImage(wall.getWall
   Name());
246.                     wall.setLogo(mediaWallSrv.saveWallImage(w
   all.getWallName(), logo));
247.                 }
248.                 wall = mediaWallSrv.updateWall(wall);
249.                 retVal = Response.ok(wall).status(Status.OK).
   build();
250.             }
251.
252.             return retVal;
253.         }
254.
255.         /**
256.          * This method is used to delete logo from Media Wall
   . It takes passed
257.          * parameter id to find wall and delete its logo.
258.          *
259.          * @param id
260.          *            This is MediaWall's ID.
261.          * @return Success: This returns Media Wall with its
   updated information and (Status code: OK).
262.          *         Failure: This returns (Status code: NOT_FO
   UND).
263.          */
264.         @DELETE
265.         @Path("/{id}/logo")
266.         public Response deleteLogoFromWall(@PathParam("id") I
   nteger id) {
267.             Response retVal;
268.
269.             MediaWall wall = mediaWallSrv.getWallById(id);
270.             if (wall != null) {
271.                 if (wall.getLogo() == null && wall.getLogo().
   isEmpty()) {
272.                     retVal = Response.status(Status.NOT_FOUND
   ).build();
273.                 } else {
274.                     mediaWallSrv.deleteWallImage(wall.getWall
   Name());
275.                     InputStream logo = getClass().getResource
   AsStream("/images/default.jpg");
```

```
276.                     wall.setLogo(mediaWallSrv.saveWallImage(w
   all.getWallName(), logo));
277.                     retVal = Response.ok(wall).status(Status.
   OK).build();
278.                 }
279.                 mediaWallSrv.updateWall(wall);
280.             } else {
281.                 retVal = Response.status(Status.NOT_FOUND).bu
   ild();
282.             }
283.             return retVal;
284.         }
285.     }
```

## 8.2. Daļa no Mediju sienas datu piekļuves objekta

```
1.  @Service
2.  public class MediaWallDAOImpl implements MediaWallDAO{
3.
4.  @Autowired
5.  MediaWallRepository mediaWallRepo;
6.
7.  /**
8.   * This method saves requested data to the database.
9.   *
10.  * @param entity MediaWall's object
11.  *
12.  * @return MediaWall Returns saved media wall.
13.  */
14. @Override
15. public MediaWall save(@Valid MediaWall entity) {
16.     return mediaWallRepo.save(entity);
17. }
18.
19. /**
20.  * This method deletes requested MediaWall from the database.
21.  *
22.  * @param id MediaWall's id
23.  */
24. @Override
25. public void delete(Integer id) {
26.     mediaWallRepo.delete(id);
27. }
28.
29. /**
30.  * This method returns all found media walls from database
31.  *
32.  * @return List Returns all existing media walls from database.
33.  */
34. @Override
```

```
35.  public List<MediaWall> findAll() {
36.      return mediaWallRepo.queryAll();
37.  }
```

### 8.3. Daļa no Mediju sienas servisa

```
1.  @Service
2.  ublic class MediaWallServiceImpl implements MediaWallService {
3.
4.  private static final Logger logger = LoggerFactory.getLogger(MediaWallServiceImpl
    .class);
5.
6.  @Autowired
7.  MediaWallDAO mediaWallDAO;
8.
9.  /**
10.  * This method calls database layer method to create new MediaWall in
11.  * database.
12.  *
13.  * @param wall
14.  *            MediaWall's data.
15.  *
16.  * @return wall Created MediaWall's data.
17.  */
18. @Override
19. public MediaWall createWall(@Valid MediaWall wall) {
20.     return this.saveWall(wall);
21. }
22.
23. /**
24.  * This method calls database layer method to save data to database.
25.  *
26.  * @param wall
27.  *            MediaWall's data.
28.  *
29.  * @return wall Saved MediaWall's data.
30.  */
31. @Override
32. public MediaWall saveWall(@Valid MediaWall wall) {
33.     MediaWall retVal = null;
34.     try {
35.         retVal = mediaWallDAO.save(wall);
36.     } catch (Exception e) {
37.         logger.info("saveWall!", e);
38.     }
39.     return retVal;
40. }
41.
42. /**
43.  * This method calls database layer method to delete data from database.
44.  *
45.  * @param id
46.  *            MediaWall's id.
47.  */
48. @Override
49. public void deleteWall(Integer id) {
50.     try {
```

```
51.         mediaWallDAO.delete(id);
52.     } catch (Exception e) {
53.         logger.info("deleteWallID!", e);
54.     }
55. }
56. 
57. /**
58.  * This method calls database layer method to get all MediaWalls from
59.  * database.
60.  *
61.  * @return List All existing MediaWalls.
62.  */
63. @Override
64. public List<MediaWall> getAllWalls() {
65.     List<MediaWall> retVal = null;
66.     try {
67.         retVal = mediaWallDAO.findAll();
68.     } catch (Exception e) {
69.         logger.info("getAllWalls!", e);
70.     }
71.     return retVal;
72. }
```

### 8.4. Mediju sienas modelis

```
1. @Entity
2. @Table(name = "MediaWall")
3. public class MediaWall extends BaseClass implements Serializabl
   e {
4. 
5.     private static final long serialVersionUID = 1L;
6. 
7.     // Field that holds MediaWall's picture.
8.     @Column(name = "LOGO", length = 255)
9.     private String logo = null;
10. 
11.         // Field that holds MediaWall's name.
12.         @NotNull(message="WallName field cannot be empty.")
13.         @Size(min = 1, max = 32, message="Your wallName shoul
   d be between 1 - 32 characters.")
14.         @Column(name = "WALL_NAME", unique=true, length = 32,
    nullable = false)
15.         private String wallName = null;
16. 
17.         // Field that holds MediaWall's search parameters.
18.         @PrivateOwned
19.         @OneToMany(cascade = CascadeType.ALL, fetch = FetchTy
   pe.EAGER)
20.         @JoinColumn(name = "MEDIAWALL_ID", referencedColumnNa
   me = "ID")
21.         @NotEmpty(message="Search parameters cannot be empty.
   ")
```

```
22.          @Valid
23.          private List<SearchInfo> searchInfo = null;
24.
25.          // Field that holds MediaWall's posts.
26.          @PrivateOwned
27.          @OneToMany(cascade = CascadeType.ALL, fetch = FetchTy
   pe.EAGER, mappedBy="wall")
28.          @Valid
29.          @JsonManagedReference
30.          @OrderBy("createdAt DESC")
31.          private List<TwitterPosts> twitterPosts = null;
32.
33.          public MediaWall() {
34.              // Do nothing because of JPA
35.          }
36.
37.          public MediaWall(String logo, String wallName) {
38.              this.logo = logo;
39.              this.wallName = wallName;
40.          }
41.
42.          public MediaWall(String logo, String wallName, List<S
   earchInfo> searchInfo) {
43.              this.logo = logo;
44.              this.wallName = wallName;
45.              this.searchInfo = searchInfo;
46.          }
47.
48.          public MediaWall(String logo, String wallName, List<S
   earchInfo> searchInfo, List<TwitterPosts> twitterPosts) {
49.              this.logo = logo;
50.              this.wallName = wallName;
51.              this.searchInfo = searchInfo;
52.              this.twitterPosts = twitterPosts;
53.          }
54.
55.          // Getter for MediaWall logo.
56.          public String getLogo() {
57.              return logo;
58.          }
59.
60.          // Setter for MediaWall logo
61.          public void setLogo(String param) {
62.              this.logo = param;
63.          }
64.
65.          // Getter for MediaWall name.
66.          public String getWallName() {
67.              return wallName;
68.          }
69.
70.          // Setter for MediaWall name.
```

```
71.          public void setWallName(String param) {
72.              this.wallName = param;
73.          }
74.
75.          // Getter for MediaWall search parameters.
76.          public List<SearchInfo> getSearchInfo() {
77.              return searchInfo;
78.          }
79.
80.          // Setter for MediaWall search parameters.
81.          public void setSearchInfo(List<SearchInfo> param) {
82.              this.searchInfo = param;
83.          }
84.
85.          // Getter for MediaWall posts.
86.          public List<TwitterPosts> getTwitterPosts() {
87.              return twitterPosts;
88.          }
89.
90.          // Setter for MediaWall posts.
91.          public void setTwitterPosts(List<TwitterPosts> twitte
   rPosts) {
92.              this.twitterPosts = twitterPosts;
93.          }
94.
95.          /**
96.           * Method that merges two walls into one replacing ol
   d data with new one.
97.           *
98.           * @param newWall MediaWall object.
99.           */
100.         public void mergeWall(MediaWall newWall){
101.             this.wallName = newWall.wallName;
102.             this.logo = newWall.logo;
103.             if(newWall.searchInfo != null && !(this.searchInf
   o.equals(newWall.searchInfo))){
104.                 this.searchInfo = newWall.searchInfo;
105.                 this.twitterPosts = null;
106.             }
107.         }
108.
109.         @Override
110.         public String toString() {
111.             return "MediaWall [logo=" + logo + ", wallName="
   + wallName + ", searchInfo="
112.                     + searchInfo + ", twitterPosts=" + twitte
   rPosts + "]";
113.         }
114.     }
```

# 9. SECINĀJUMI

Izstrādājot tīmekļa vietni tika iegūta pieredze un zināšanas gan programmēšanas valodā *Java,* gan tehnoloģijās kā *HTML, CSS* un *JavaScript*. Būtisku daļu laika no izstrādes prasīja *Spring* satvara apgūšana, ar kura palīdzību tika izstrādāta daļa programmatūras produkta..

Kvalifikācijas darba rezultāts ir funkcionējoša tīmekļa vietne, taču laika trūkuma dēļ neizdevās izveidot lietotāju draudzīgu administratora paneli, tāpēc autentificētam lietotājam izveidot mediju sienu ir iespējams caur paplašinājumiem, kuri spēj darboties ar REST API un veikt dažādus HTTP pieprasījumus.

Izstrādāto programmatūru varētu uzlabot pievienojot iespēju iegūt datus no Instagram un Facebook vietnēm, kā arī izveidot paredzēto administratora paneli, kuram būtu iespēja pārskatīt katras izveidotās mediju sienas sīkziņas un manuāli izvēlēties, kuras neattēlot tīmekļa vietnē.

# 10. IZMANTOTĀ LITERATŪRA


1. Bootstrap dokumentācija [tiešsaiste]. - [atsauce 20.05.2016.]. Pieejams: http://getbootstrap.com/
2. Java Persistence [tiešsaite]. – [atsauce 20.05.2016.]. Pieejams: https://en.wikibooks.org/wiki/Java_Persistence
3. **LVS 72:1996** - Ieteicamā prakse programmatūras projektējuma aprakstīšanai.
4. **LVS 68:1996** - Programmatūras prasību specifikācijas ceļvedis;
5. RESTful tīmekļa servisu arhitektūras vadlīnijas [tiešsaiste]. – [atsauce 20.05.2016.]. Pieejams:http://blog.mwaysolutions.com/2014/06/05/10-best-practices-for-better-restful-api/
6. SAP HANA Cloud dokumentācija [tiešsaiste]. – [atsauce 20.05.2016.]. Pieejams: https://help.hana.ondemand.com/help/frameset.htm?e9137493bb57101492c6858c8d6b0b62.html
7. Spring dokumentācija [tiešsaiste] – [atsauce 20.05.2016.]. Pieejams: http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/
8. Twitter dokumentācija [tiešsaiste]. – [atsauce 20.05.2016.]. Pieejams:https://dev.twitter.com/rest/public

Kvalifikācijas darbs „***Sociālo mediju ziņojumu plūsmu pārskata vietne***” 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: ***Rihards Gailis*** **_____ ___.05.2016**.

Rekomendēju darbu aizstāvēšanai
Darba vadītājs: ***Mg.sc. ing Toms Mediņš*** **_____ ___.05.2016.**

Recenzents: ***M.dat. Maija Ļaksa***

Darbs iesniegts 30.05.2016.
Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2016. prot. Nr. _____

Komisijas sekretārs(-e): _____