LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

# “Microblog”

OBJEKTORIENTĒTAS PROGRAMMĒŠANAS
PROJEKTA DOKUMENTĀCIJA

Autors: **Ņikita Kruglovs, Artjoms Strelkovs**
Studenta apliecības Nr.: nk21087, as21264
Darba vadītājs: Dr.sc.comp. Valdis Vītoliņš

RĪGA 2024

# ANOTĀCIJA

Microblog ir tīmekļa lietotne, kas ļauj lietotājiem veidot un dzēst ziņas, komentēt, veidot reakcijas, sekot citiem lietotājiem un pārvaldīt savus profilus. Sistēma ir izstrādāta, lai nodrošinātu vienkāršu un intuitīvu lietotāja pieredzi ar funkcionālām iespējām, kas nepieciešamas mazam blogam. Projekts ietver vairākus komponentus, tostarp lietotāju autentifikāciju, ziņu pārvaldību un interaktīvas funkcijas.

# ABSTRACT

Microblog is a web application that allows users to create, edit, and delete posts, comment, react, follow other users, and manage their profiles. The system is designed to provide a simple and intuitive user experience with the necessary functional capabilities for a small blog. The project includes several components, including user authentication, post management, and interactive features.

# SATURA RĀDĪTĀJS

# IEVADS

Mūsdienās sociālie tīkli un emuāri ir neatņemama interneta sastāvdaļa, kur cilvēki dalās ar savām domām, pieredzi un informāciju. Microblog projekts ir izstrādāts, lai nodrošinātu lietotājiem ērtu platformu, kurā viņi var veidot un pārvaldīt savus blogus, kā arī mijiedarboties ar citiem lietotājiem, izmantojot komentārus un reakcijas.

# NOLŪKS

Šī projekta nolūks ir izstrādāt pilnvērtīgu tīmekļa lietotni, kas nodrošina lietotājiem iespēju veidot, rediģēt un dzēst ziņas, komentēt un veidot reakcijas, sekot citiem lietotājiem un pārvaldīt savus profilus. Projekts ir vērsts uz lietotāja pieredzes uzlabošanu, nodrošinot vienkāršu un intuitīvu saskarni, kas ir viegli lietojama gan jauniem, gan pieredzējušiem lietotājiem.

## Darbības sfēra

Microblog projekts ietver sekojošas funkcionalitātes un komponentes:

- Lietotāju reģistrācija un autentifikācija: Lietotāji var reģistrēties, pieteikties un pārvaldīt savus kontus.
- Ziņu pārvaldība: Lietotāji var veidot un dzēst ziņas.
- Komentāru un reakciju pārvaldība: Lietotāji var pievienot komentārus un reaģēt uz ziņām.
- Sekotāju pārvaldība: Lietotāji var sekot citiem lietotājiem un pārvaldīt savus sekotājus.
- Profila pārvaldība: Lietotāji var atjaunināt savu profila informāciju, tostarp profila attēlu un citas personiskās detaļas.
- Meklēšana: Lietotāji var meklēt citus lietotājus pēc vārda.

Projekta darbības sfēra ir nodrošināt stabilu un drošu platformu, kas atbalsta augstas veiktspējas prasības un ir pieejama 24/7 ar minimālām dīkstāvēm. Tas ietver arī datu drošības un privātuma aizsardzības mehānismus, lai nodrošinātu lietotāju datu integritāti un konfidencialitāti.

# 1. VISPĀRĒJAIS APRAKSTS

## 1.1.Esoša stāvokļa apraksts

Microblog ir tīmekļa lietotne, kas izstrādāta, lai sniegtu lietotājiem platformu ziņu, komentāru un reakciju veidošanai, kā arī lietotāju mijiedarbībai. Sistēma ir paredzēta, lai nodrošinātu vienkāršu un intuitīvu lietotāja pieredzi ar dažādām sociālajām funkcijām.

## 1.2.Pasūtītājs

Projekts ir izstrādāts kā studenta iniciatīva kvalifikācijas darba ietvaros.

## 1.3.Produkta perspektīva

Sistēma nodrošina lietotājiem iespēju veidot, rediģēt un dzēst ziņas, kā arī veidot komentārus un reakcijas. Tā ietver arī lietotāju autentifikācijas un autorizācijas mehānismus, lai nodrošinātu drošību un datu aizsardzību.

## 1.4.Darījumprasības

- Lietotāji var veidot, rediģēt un dzēst ziņas.
- Lietotāji var komentēt un reaģēt uz ziņām.
- Lietotāji var sekot citiem lietotājiem un pārvaldīt savus sekotājus.
- Sistēma nodrošina lietotāju autentifikāciju un autorizāciju.

## 1.5.Sistēmas lietotāji

### 1.5.1. Lietotāju apraksts

Uz doto brīdi tiek izmantotās 2 lietotāju grupas ar savām iespējam un uzdevumiem, kas ir atspoguļots tabulā *(sk. 1.1. tabula)*.

*1.1. tabula*

**Lietotāju iespēju un ierobežojumu tabula**

| Grupas nosaukums | Iespējas | Procesa ierobežojumi |
|---|---|---|
| Neautorizēts lietotājs | Var reģistrēties sistēma. Izmantojot izveidoto lietotāju, var ielogoties. | Nevar izmantot mājaslapas funkcionalitāti. |
| Autorizēts lietotājs | Var reģistrēties sistēma. Izmantojot izveidoto lietotāju, var ielogoties. Var rediģēt profilu, datus. Veidot un dzēst savus postus. Veidot un dzēst komentārus. Veidot reakcijas. Meklēt lietotājus. Pierakstīties, atrakstīties no citiem. Bloķēt citus lietotājus. Dzēst savu profilu. | Var lietot sistēmu pilnā apmērā. |

## 1.6. Vispārējie ierobežojumi

- Interneta pieslēgums ir nepieciešams, lai izmantotu sistēmu.
- Piekļuve projektam.
- Piekļuve datu bāzes datiem.

## 1.7.Pieņēmumi un atkarības

### 1.7.1. Pieņēmumi

- Programmatūra atrodas uz lokālas mašīnas, vai datora.
- Ir izmantots SQL Server, lai darboties ar datu bāzēm.

# 2. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

## 2.1. Funkcionālās Prasības

2.1.1. Sistēma nodrošina lietotāju reģistrāciju un autentifikāciju.

2.1.2. Lietotāji var veidot un dzēst ziņas.

2.1.3. Lietotāji var komentēt un reaģēt uz ziņām.

2.1.4. Lietotāji var sekot citiem lietotājiem un pārvaldīt sekotājus.

2.1.5. Lietotāji var pierakstīties uz cita lietotāja ziņām un atrakstīties.

2.1.6. Lietotāji var bloķēt citus lietotājus.

2.1.7. Lietotāji var mainīt savus datus.

## 2.2.Nefunkcionālās prasības

### 2.2.1. Veiktspējas prasības

1. Sistēma spēj apkalpot līdz 100 vienlaicīgiem lietotājiem..

### 2.2.2. Drošība

1. Pieslēgšana lietotnei ir iespējama tikai ar individuālo loginu un paroli
2. Sistēma izmanto šifrēšanu, lai aizsargātu lietotāju datus.
3. Tiek nodrošināta lietotāju autentifikācija un autorizācija.

### 2.2.3. Pieejamības prasības

1. Sistēma ir pieejama tikai, kad projekts ir instalēts un iedarbināts.

### 2.2.4. Izmantojamība

1. Sistēmai ir jābūt lietotājam draudzīgai un viegli lietojamai.
2. Sistēmai jānodrošina intuitīvs interfeiss, kas ļauj lietotājiem viegli veidot un dzēst ziņas, komentārus un profilus.

### 2.2.5. Uzturēšanas prasības

1. Sistēmai jābūt viegli uzturamai un atjaunojamai.
2. Sistēmai jābūt dokumentētai, lai izstrādātāji varētu viegli izprast un veikt izmaiņas sistēmā.
3. Sistēmai jāatbalsta vienkārša kļūdu novēršana un sistēmas atjaunināšana ar minimālu dīkstāvi..

### 2.2.6. Uzticamības prasības

1. Sistēmai jābūt stabilai un uzticamai ar minimālu kļūdu daudzumu.
2. Sistēmai jānodrošina datu integritāte un konfidencialitāte.
3. Sistēmai jābūt izstrādātai tā, lai novērstu datu zudumu un nodrošinātu datu atjaunošanu ārkārtas situācijās.

# 4. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

## 4.1. Sistēmas arhitektūra

Microblog projekta arhitektūra ir balstīta uz 3 līmeņu arhitektūras modeli, kas ietver "Data", "Domain" un "Presentation" līmeņus. Šāda arhitektūras pieeja nodrošina skaidru atdalīšanu starp datu piekļuvi, biznesa loģiku un lietotāja saskarni, veicinot labāku kodu organizāciju, uzturamību un atkārtotu izmantojamību.

### 4.1.1. Data līmenis

"Data" līmenis atbild par datu glabāšanu un piekļuvi tiem. Tas ietver datu bāzes kontekstu, entītijas, kā arī migrācijas un datu piekļuves loģiku.

Sastāvdaļas:

**Entities:** Datu modeļi, kas atbilst datu bāzes tabulām (piemēram, Post, Comment, Reaction, Subscription, Block, Image).

**DbContext:** Datu bāzes konteksts, kas nodrošina savienojumu ar datu bāzi un entītiju pārvaldību (piemēram, ApplicationDbContext).

**Repositories:** Datu piekļuves slānis, kas ietver metodes datu nolasīšanai, pievienošanai, atjaunināšanai un dzēšanai no datu bāzes (PostRepository, CommentRepository, ReactionRepository, SubscriptionRepository, BlockRepository, ImageRepository).

Piemērs:

**PostRepository:** Pārvalda ziņu (Post) objektu piekļuvi un manipulācijas datu bāzē.

### 4.1.2. Domain līmenis

"Domain" līmenis atbild par biznesa loģiku un noteikumiem, kas nosaka, kā sistēma darbojas. Tas ietver servisu slāni, kurā ir definētas galvenās biznesa loģikas metodes un interfeisi.

Sastāvdaļas:

**Models:** Domēna modeļi, kas atspoguļo biznesa objektus un to loģiku (piemēram, Post, Comment ar papildus biznesa loģiku).

**Services:** Servisu slānis, kas ietver biznesa loģiku un nodrošina mijiedarbību starp Data un Presentation līmeņiem (PostService, CommentService, ReactionService, UserLoggingService).

**Interfaces:** Interfeisi, kas definē metodes, kuras implementē servisi un repositoriji (piemēram, IPostService, ICommentService, IReactionService).

**Piemērs:**

PostService: Pārvalda loģiku, kas saistīta ar ziņu veidošanu, rediģēšanu un dzēšanu, kā arī lietotāju ziņu ielādes funkcionalitāti.

### 4.1.3. Presentation līmenis

"Presentation" līmenis atbild par lietotāja saskarni un mijiedarbību ar lietotāju. Tas ietver tīmekļa lietotnes komponentes, piemēram, kontrolierus, skatus un klienta puses skriptus.

Sastāvdaļas:

**Controllers**: Kontrolieri, kas apstrādā HTTP pieprasījumus, mijiedarbojas ar domēna servisiem un atgriež atbildes (PostController, CommentController, UserController, ReactionController).

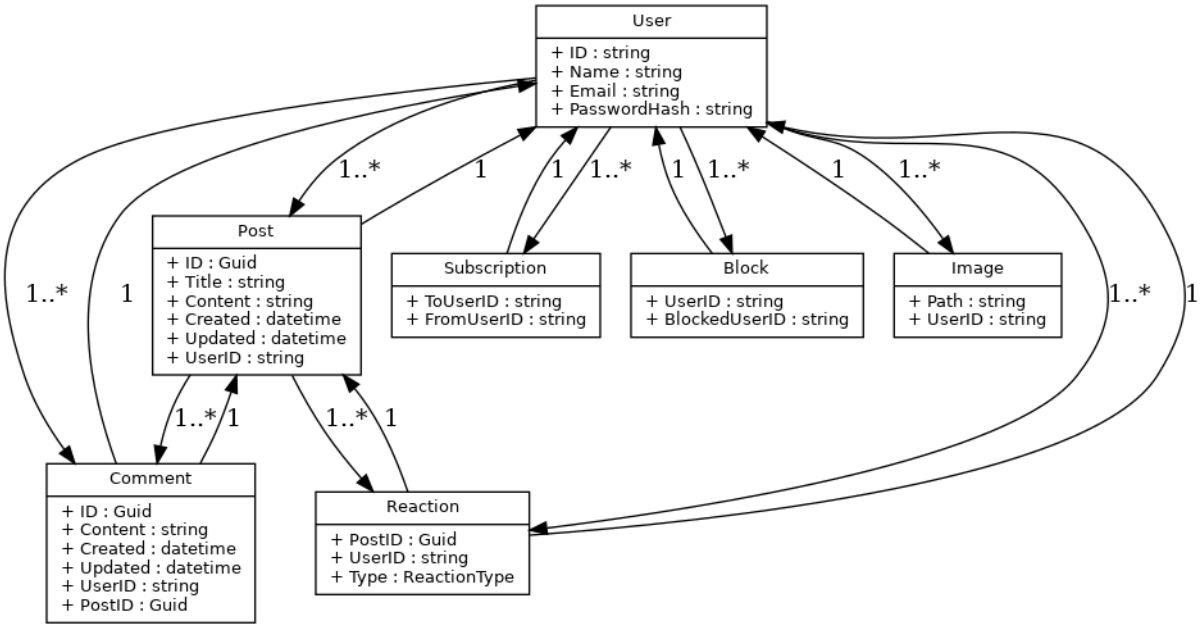
**Views**: Skati, kas nodrošina HTML, CSS un JavaScript, lai renderētu lietotāja saskarni.

**ViewModels**: Modeļi, kas tiek izmantoti, lai pārsūtītu datus starp kontrolieriem un skatiem.

Piemērs:

**PostController**: Apstrādā pieprasījumus, kas saistīti ar ziņām, un izmanto PostService, lai veiktu nepieciešamās darbības, piemēram, jaunas ziņas izveidi vai esošās ziņas dzēšanu.

## 4.2. Klašu diagramma




*Ilustrācija 1.1. Klašu diagramma*

Klašu diagramma apraksta galvenās Microblog sistēmas klases un to attiecības. Ir iekļautas šādas klases:

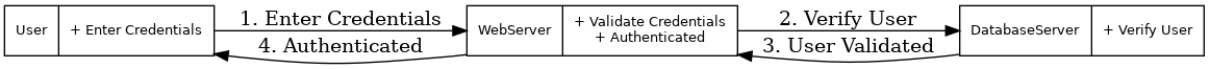
- **User**: Satur informāciju par lietotāju, tostarp ID, vārdu, e-pasta adresi un paroles hešu.
- **Post**: Satur informāciju par ziņām, tostarp ID, virsrakstu, saturu, izveides datumu, atjaunināšanas datumu un lietotāja ID.
- **Comment**: Satur informāciju par komentāriem, tostarp ID, saturu, izveides datumu, atjaunināšanas datumu, lietotāja ID un ziņas ID.

- **Reaction**: Satur informāciju par reakcijām uz ziņām, tostarp ziņas ID, lietotāja ID un reakcijas veidu.
- **Subscription**: Satur informāciju par abonementiem, tostarp abonētā lietotāja ID un lietotāja ID, kurš abonē.
- **Block**: Satur informāciju par bloķētajiem lietotājiem, tostarp lietotāja ID un bloķētā lietotāja ID.
- **Image**: Satur informāciju par attēla ceļu un lietotāja ID.

Savstarpējās attiecības starp klasēm:

- Lietotājam var būt vairāki ziņojumi, komentāri, reakcijas, abonementi, bloķējumi un attēli.
- Ziņai var būt vairāki komentāri un reakcijas.
- Komentārs un reakcija ir saistīti ar konkrētu ziņu un lietotāju.
- Abonements un bloķēšana ir saistīti ar lietotāju.

## 4.3. Secības diagramma lietotāja autentifikācijai

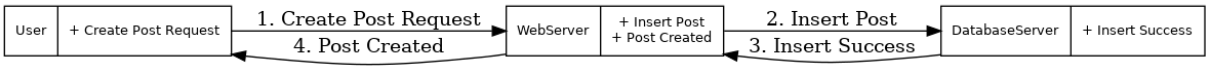



*Ilustrācija 2. Lietotāja autentifikācijas secību diagramma.*

Secības diagramma parāda jaunas ziņas izveides procesu:

1. Lietotājs nosūt pieprasījumu izveidot jaunu ziņu.
2. Tīmekļa serveris apstrādā pieprasījumu un nosūt komandu ievietot jauno ziņu datu bāzē.
3. Datu bāze apstiprina veiksmīgu ziņas ievietošanu.
4. Tīmekļa serveris atgriež apstiprinājumu lietotājam par ziņas izveidi.

## 4.1.Secības diagramma ziņas izveidei




*Ilustrācija 3. Ziņas izveidošanas secību diagramma.*

Secības diagramma parāda lietotāja autentifikācijas procesu:

1. Lietotājs ievada savus akreditācijas datus.
2. Tīmekļa serveris pārsūt akreditācijas datus uz pārbaudi datu bāzē.
3. Datu bāze pārbauda lietotāju un apstiprina tā autentiskumu.
4. Tīmekļa serveris atgriež lietotājam apstiprinājumu par autentifikāciju.

# 5. PROGRAMMATŪRAS TESTĒŠANA

## 5.1. Testpiemēru projektējums

Testpiemēru projektējums ietver dažādu sistēmas komponentu testēšanu, izmantojot vienību testus. Testēšana tika veikta, lai nodrošinātu, ka katrs atsevišķs komponents darbojas pareizi un atbilst prasībām.

**Testētie komponenti**

1. **Repozitoriji**
    - **PostRepository**: Testēta ziņu pievienošana, atjaunināšana, dzēšana un iegūšana pēc ID.
2. **Servisi**
    - **PostService**: Testēta ziņu iegūšana pēc lietotāja un abonementiem, kā arī reakciju un lietotāju informācijas iegūšana.
    - **CommentService**: Testēta komentāru iegūšana un pievienošana.
    - **UserLoggingService**: Testēta lietotāju pieslēgšanās reģistrēšana un logu atiestatīšana.

**Testpiemēru projektēšanas process**

Testpiemēru projektēšanas procesā tika izmantotas šādas metodes:

- **Mokešana (Mocking)**: Lai simulētu datu bāzes un citu ārējo atkarību uzvedību, tika izmantoti mokēšanas rīki, piemēram, Moq.
- **Vienību testi (Unit Tests)**: Izstrādāti testi katram atsevišķam komponentam, lai pārbaudītu tā funkcionalitāti izolēti no pārējās sistēmas.
- **Testēšanas ietvars (xUnit)**: Testēšanai tika izmantots xUnit ietvars, kas nodrošina strukturētu pieeju testu rakstīšanai un izpildei.

# 6. PROJEKTA ORGANIZĀCIJA

Projekts Microblog tika izstrādāts divu cilvēku komandas ietvaros. Mēs kopīgi izstrādājām sistēmas konceptu, plānojām dizainu, izstrādājām backend un frontend komponentes, aizpildījām datu bāzi un sagatavojām dokumentāciju. Komandas sadarbība nodrošināja vienotu pieeju projekta izstrādei un veiksmīgu tā realizāciju. Mūsu komandas pienākumi bija sadalīti sekojoši:

- Sistēmas dizains: Kopīgi izstrādājām sistēmas arhitektūru un plānojām dizainu, lai nodrošinātu vienotu un efektīvu sistēmas darbību.
- Backend izstrāde: Izstrādājām servera puses loģiku, datu bāzes struktūru un API, lai nodrošinātu stabilu un drošu datu apstrādi.
- Frontend izstrāde: Veidojām lietotāja saskarni, kas ir intuitīva un viegli lietojama, nodrošinot ērtu piekļuvi visām sistēmas funkcijām.
- Datu bāzes aizpildīšana: Aizpildījām datu bāzi ar nepieciešamajiem sākotnējiem datiem, lai sistēma varētu darboties pilnvērtīgi.
- Dokumentācijas sagatavošana: Sagatavojām nepieciešamo dokumentāciju, kas apraksta sistēmas darbību, uzbūvi un lietošanu..

## 6.1. Kvalitātes nodrošināšana

Projekta kvalitātes nodrošināšanas procesā mēs pārbaudījām sistēmu uz kļūdām, veicām veiktspējas testēšanu un realizējām iespējami daudz funkcionalitātes, cik tas bija iespējams noteiktajā laikā. Kvalitātes nodrošināšanas mērķis bija nodrošināt stabilu un lietotājam draudzīgu sistēmu.

## 6.2. Konfigurāciju pārvaldība

Darba kods glabājas uz GIT krātuves un visas izmaiņas ir piefiksētas tur. Lai pārvaldītu kodu, tas tika klonēts uz konkrēto datoru un projekts bija uzstādīts pēc instrukcijas.