Latvijas Republikas izglītības un zinātnes ministrija

Profesionālās izglītības kompetences centrs

Daugavpils Tehnoloģiju un tūrisma tehnikums

**PRAKTISKAIS DARBS**

Programmēšanas tehniķis

specialitāte

1. projekta darbs

temats

Testēšanas sistēmas izveidošana

PASKAIDROJOŠAIS RAKSTS

**DTTT.P.D.0024.0024PA**

Profesijas kods 33 481 031 Programmēšanas nodaļa

PR-21. grupa

Izveidoja A.Rinkeviča

paraksts V.,Uzvārds

Pārbaudīja

paraksts V.,Uzvārds

2023./2024. m.g.

**Saturs**

[Ievads 3](#_Toc169569110)

[1. Uzdevuma nostādne 4](#_Toc169569111)

[1.1. Sistēmas modelis 4](#_Toc169569112)

[1.1.1. Priekšmetiskās jomas informācijas modelis 4](#_Toc169569113)

[1.1.2. Sistēmas apkārtnes shēma 5](#_Toc169569114)

[1.1.3. Viedokļu analīze 5](#_Toc169569115)

[1.1.4. Datu modelēšana 6](#_Toc169569116)

[1.1.5. Datu vārdnīca 7](#_Toc169569117)

[1.2. Sistēmas evolūcija 7](#_Toc169569118)

[1.3. Sistēmas funkcionālās prasības 8](#_Toc169569119)

[1.4. Sistēmas nefunkcionālās prasības 11](#_Toc169569120)

[2. Priekšmetiskās jomas klašu sistēmas izstrāde 15](#_Toc169569121)

[2.1. Klašu diagramma 15](#_Toc169569122)

[2.2. Klašu realizēšana Java valodā 17](#_Toc169569123)

[3. Testa programmas izstrāde 39](#_Toc169569124)

[3.1. Testēšanas metodikas 39](#_Toc169569125)

[3.2. Testēšanas programmas struktūra 39](#_Toc169569126)

[3.3. Testēšanas rezultāti 44](#_Toc169569127)

[4. Lietotāja rokasgrāmata 45](#_Toc169569128)

[4.1. Lietotāja instrukcija 45](#_Toc169569129)

[4.2. Palīdzības sistēma (Help) 45](#_Toc169569130)

[Secinājumi 47](#_Toc169569131)

[Izmantotās literatūras (informācijas avotu) saraksts 48](#_Toc169569132)

[Pielikums A. Klašu sistēma 49](#_Toc169569133)

[Pielikums B. testa programmas kods 58](#_Toc169569134)

# Ievads

Projekta darbā ir aprakstīta testēšanas programma, kura izpilda sekojošas funkcijas:

* lietotāju ielogošana;
* lietotāju reģistrēšana;
* datu ievade;
* datu saglabāšana;
* datu rediģēšana;
* testa izveidošana;
* testa saglabāšana;
* testa rediģēšana;
* testa pildīšana;
* rezultātu aprēķināšana.
* rezultātu izvade;

Sistēma nodrošina audzēkņa testēšanu par noteiktu tēmu/vielu, tas ir domāts, lai skolotās varētu viegli izlikt atzīmi skolēnam digitālā veidā attālināti vai klātienē.

Programmas saskarne ļauj viegli izveidot testus un jautājumus tiem. Katram lietotājam – skolēnam un skolotājam – ir savas iespējas.

Skolēns varēs apskatīt sev pieejamo testu sarakstu, veikt testus un apskatīt rezultātus, bet skolotājs varēs pievienot, rediģēt, kā arī dzēst testus.

# Uzdevuma nostādne

## Sistēmas modelis

### Priekšmetiskās jomas informācijas modelis

Apskatāmā sistēmā tika izdalīti objekti: Datu bāze (Fails ar atbildēm), Tests, Rezultāti, Administrators, Skolotājs, Lietotājs. Šie objekti un saites starp tiem ir paradīti (1.1. att.).

Datu bāze ar rezultātiem

Tests

Rezultāti

Skolēns

Skolotājs

Datu bāze ar lietotāja datiem

att. 1.1. Vienkāršots testēšanas sistēmas modelis

Saišu nozīme vienkāršotā nodaļas modelī:

* Skolotājs ir testa veidotājs un pārbaudītājs, kuram ir pieeja pie atbildēm, testam un lietotāja rezultātiem;
* Datu bāze satur failus ar jautājumiem un ar pariezām atbildēm, kā arī ar lietotāju datiem;
* Skolēns ir testa pildītājs, kas beigās var apskatīt savus rezultātus, nepieciešamības gadījumā viņš varēs apskatīt savas izvēlētās atbildes;
* Tests ir instruments, kas palīdzēs pārbaudīt lietotāju zināšanas un novērtēt to ar atzīmi. Pēc testa beigām vairs nebūs iespējams atkārtot testu, ja skolotājs to vēlēsies;
* Rezultāti parāda atzīmi, pareizās un nepareizās atbildes skaitliski un procentuāli. Ja skolēns vēlas, viņš var jebkurā laikā apskatīt savus rezultātus īpašā logā.

### Sistēmas apkārtnes shēma

Dati, kā atbildes, ko ievada audzēknis tiek saglabāti failā. Tomēr dažreiz atbildes netiek automātiski novērtētas un skolotājam ir iespēja manuāli novērtēt iesniegto atbildi. Saskarni ar lietotājiem nodrošinās interfeisa modulis (1.2. att.).

Lietotājs

Lietotāja saskarne

Sistēma

Datu bāze

Lietotāja dati un rezultāti

att. 1.2. Sistēmas apkārtnes modeļa piemērs

### Viedokļu analīze

Savāktie un identificētie viedokļi par sistēmu ir attēloti ar burbuļdiagrammas palīdzību (1.3. att.), tas faktiski ir viedokļu kopums, kur katra viedokļa nosaukums ir ierakstīts atsevišķa elipsē.

Nefunkcionālie viedokļi

Funkcionālie viedokļi

Datu viedokļi

Viedokļi par lietotāju

att. 1.3. Viedokļu burbuļu diagramma

Otrais etaps viedokļu analīzē ir viedokļu klasifikācija, kad viedokļi ir sagrupēti grupās (1.4. att.).

att. 1.4. Viedokļu hierarhiskā struktūra

### Datu modelēšana

Datu modelēšanai ir tiek izmantots ER modelis. Testēšanas sistēmas ER modelis (1.5. att.) ietver datu entītiju kopumu un relāciju kopumu starp datu entītijām, pie kam ar šo modeli var attēlot dažāda tipa relācijas starp datu elementiem.

Skolēns

Pilda

Tests

Satur

Jautājums

Izveido

Skolotājs

Dzēst

Rediģēt

att. 1.5. Vienkāršots testēšanas sistēmas ER modelis

### Datu vārdnīca

Lietotāja dati

|  |  |  |
| --- | --- | --- |
| Datu tips | Nosaukums | Apraksts |
| int | **ID** | Lietotāja kārtas numurs |
| String | **login** | Lietotājs ievada savu lietotājvārdu |
| String | **password** | Lietotājs ievada savu paroli |
| String | **firstName** | Lietotājs ievada savu vārdu |
| String | **lastName** | Lietotājs ievada savu uzvārdu |
| String | **eMail** | Lietotājs ievada savu pastu |
| boolean | **rights** | Lietotāja tiesības. Skolēns var tikai veikt testus un redzēt rezultātus (false), bet skolotājs var pievienot, rediģēt un dzēst testus (true) |
| Int[] | **pointList** | Cik punktus lietotājs ir ieguvis testam |

Testa dati

|  |  |  |
| --- | --- | --- |
| Datu tips | Nosaukums | Apraksts |
| int | **ID** | Testa kārtas numurs |
| Int[] | **questionidlist** | Testa jautājumu skaits |
| Int[] | **questionList** | Datubāzē esošo testu saraksts |
| String | **name** | Testa nosaukums |

Jautājuma dati

|  |  |  |
| --- | --- | --- |
| Datu tips | Nosaukums | Apraksts |
| int | **ID** | Jautājuma kārtas numurs |
| String | **text** | Jautājuma apraksts |
| String[] | **answers** | Atbilžu uz jautājumu variācijas |
| int | **rightsAnswers** | Pareizo atbilžu indekss |

## Sistēmas evolūcija

Testēšana sistēma ir neatkarīga un pašpietiekama programma. Produktam nav paredzētas vairākas sastāvdaļas.

Programmai ir septiņas ārējās saskarnes, kas redzamas 1.4. nodaļā: viena sistēmas reģistrācijai, viena ielogošanai, viena lietotāja profilam, viena lietotāja datu ievadīšanai, viena testa saraksta apskatīšanai, viena testa izveidei, viens testa jautājumu izveidei un rediģēšanai, viens, lai veiktu testus, un vienu testu rezultātiem. Grafisko lietotāja saskarni var paplašināt, profila logā pievienot savu fotoattēlu. Testēšanas sistēma nodrošina divu veidu lietotājus: skolēnus un skolotājus.

Plānots pievienot vēl vienu lietotāju grupu ar funkciju veidot jaunus testus un pārbaudīt nokārtotos testus. Administratoram plānots realizēt lietotāja vārda vai paroles maiņas funkciju, kā arī pievienot un dzēst lietotājus un mainīt vīna lomas.

Programmā ir divi testi, katrā no tiem ir 10 jautājumi un daudz, bet nākotnē sistēmu var pilnveidot, pievienojot jaunas testu tēmas, parādot un izskaidrojot kļūdas pēc testa kārtošanas, vērtējot datus par lietotājiem, parādot lietotāju datus lietotāja izvēlnē vai administratora izvēlnē.

Dažiem datu veidiem sistēma izmantos datubāzē saglabātos datus (datubāzes savienojums, testa jautājumi un atbildes u. c.).

## Sistēmas funkcionālās prasības

**1. Funkcija „Reģistrēties sistēmā”**

***ID: T1***

**Ievads:** Ļauj ievadīt nepieciešamus datus, lai izveidotu savu kontu. Visi ievadīti dati tiek saglabāti failā (datu bāzē).

**Ievade:**

1. Lietotāja lietotājvārds
2. Lietotāja parole
3. Ievadītās paroles apstiprināšana.
4. Piederība vai nu “skolēnam”, vai “skolotājam”.

**Apstrāde:**

1. Lietotāja lietotājvārda saglabāšana failā.
2. Lietotāja paroles saglabāšana failā.
3. Lietotāja piederības saglabāšana.

Ja ievadītā parole ir nepareiza vai jau ir lietotājs ar šo lietotājvārdu, tad tiek atvērts kļūdas dialoglodziņš.

**Izvade:** Dialoglodziņš, kurā uzrakstīts „Reģistrēšana ir veiksmīgi pabeigta!”.

**2. Funkcija “Ielogoties sistēmā”**

***ID: T2***

**Ievads:** Ļauj ievadīt nepieciešamus datus, lai ieiet sava kontā.

**Ievade:**

1) Lietotāja lietotājvārds.

2) Lietotāja parole.

**Apstrāde:**

1) Lietotāja lietotājvārda pārbaude un atrašana DB.

2) Lietotāja paroles pārbaude un atrašana DB.

Ja ievadītā parole vai/un lietotājvārds ir nepareiza, tad tiek atvērts kļūdas dialoglodziņš.

**Izvade:** Lietotāja pārvietošana uz profila logu.

**3. Funkcija „Izveidot testu”**

**ID: *T3***

**Ievads:** Ļauj skolotājam izveidot testu un tā jautājumus.

**Ievade:**

1. Testa nosaukums.
2. Jautājumu skaits.

**Apstrāde:**

1. Testa jautājumu tabulu izveide DB;

**Izvade:** Dialoglodziņš, kurā uzrakstīts „Veiksmīgi izveidots tests!”.

**4. Funkcija „Rediģēt testu”**

**ID: *T4***

**Ievads:** Ļauj skolotājam mainīt jautājumus testā: jautājumu saturu, atbildes variācijs, atbildes pareizību.

**Ievade:**

1. Jautājumu rediģēšana.
2. Atbildes rediģēšana.
3. Jautājuma dzēšana.

Apstrāde:

1. Testa satura maiņa DB.

**Izvade:** Dialoglodziņš jautājuma satura maiņai.

**5. Funkcija “Sākt testu”**

***ID: T5***

**Ievads:** Ļauj skolotājam sākt testu.

**Ievade:** Skolotājs noklikšķina uz pogu “Sākt testu”. Pēc tam sistēma uzdod jautājumu: "Vai Jūs vēlaties sākt testu?" Skolotājaь ir divi varianti atbildēm "Jā" vai "Nē".

**Apstrāde:** Sistēma pārbauda atbildi uz uzdoto jautājumu un pēc tam sāk testu, ja uz uzdoto jautājumu atbilde "Jā". Tad tests ir pieejams lietotājiem to veikšanai.

**Izvade:** Skolotāja ekrānā tiek parādīts dialoglodziņš ar tekstu "Tests ir atļauts!"

**6. Funkcija “Pildīt testu”**

***ID: T6***

**Ievads:** Ļauj lietotājam sākt testa izpildi.

**Ievade:** Lietotājs noklikšķina uz pogu “Pildīt testu”.

**Apstrāde:** Ja tests ir pieejams izpildei, tad pēc testa izvēles tiek parādīts dialoglodziņš ar tekstu "Tests ir pieejams! Vai vēlaties izpildīt testu?". Lietotājam ir divi varianti atbildēm "Jā" vai "Nē". Ja tests nav pieejams izpildei, tad pēc testa izvēles tiek parādīts dialoglodziņš ar tekstu "Tests nav pieejams!"

**Izvade:** Lietotājā ekrānā tiek parādīts dialoglodziņš ar uzrakstu "Veiksmi testā!"

**7. Funkcija “Atbildēt uz testa jautājumu”**

***ID: T7***

**Ievads:** Ļauj ievadīt atbildi uz testa jautājumus.

**Ievade:** Lietotājs izvēlas 1 no 4 piedāvātajām atbildēm uz uzdoto jautājumu.

**Apstrāde:** Sistēma pārbauda atbildi uz jautājumu. Pēc katras pareizās atbildes sistēma pievieno 1.

**Izvade:** Pēc izvēlētās atbildes lietotājs noklikšķina uz pogas "Nākamais", pēc kura tiek parādīts nākamā izvēlne ar testa jautājumiem.

**8. Funkcija “Parādīt rezultātu”**

***ID: T8***

**Ievads:** Ļauj uzzināt nokārtotā testa rezultātus.

**Ievade:** Lietotājs noklikšķina uz pogas "Parādīt rezultātu".

**Apstrāde:** Sistēma saskaita pareizo atbilžu skaitu, nepareizo atbilžu skaitu un neatbildēto jautājuma skaitu, tad aprēķina pareizo atbilžu procentuālo vērtību, pēc tam nolasa vērtējumu par 10 balles skalu un atrod komentāru par saņemto atzīmi.

**Izvade:** Tiek atvērts dialoglodziņš, kurā tiek rakstīts šāds teksts:

1) Neatbildēto jautājuma skaits.

2) Pareizo atbilžu skaits.

3) Nepareizo atbilžu skaits.

4) Pareizo atbilžu procentuālā daļa.

5) Atzīme.

6) Sistēmas komentārs par saņemto atzīmi.

## Sistēmas nefunkcionālās prasības

1. Prasības produktam:

1.1. Lietotāju saskarne ar sistēmu notiek latviešu valodā.

1.2. Produktam nav paredzētas vairākas sastāvdaļas. Produktam ir piecas ārējās saskarnes: reģistrēšanai sistēmā, ielogošanai sistēmā, lietotāja profils, testa izveidošanai, testa pildīšanai.

1.3. Programma ir paredzēta vienam lietotājam, kurš saprot latviešu valodu. Lietotāja vecuma vai izglītības ierobežojumi nav paredzēti.

2. Ārējās saskarnes prasības:

2.1. Lietotāja saskarne: produkts paredz vienotu lietotāja saskarni.

2.2. Ekrāna formāti: minimālais ekrāna izmērs ir 10” ar minimālo izšķirtspēju –800x600 punkti.

2.3. Lietotāja saskarne “Lietotāja reģistrēšanas saskarne”. Saskarnes struktūra: skat. 1.6. att.

Skolēns

Jūs esat

Lietotājvārds:

Parole:

Parole (atkārtotu):

Reģistrēties

Ielogoties

att. 1.6. Lietotāja reģistrēšanas saskarne

Saskarnes elementi:

1. Teksta lauks “Lietotājvārds”

2. Teksta lauks ‘Parole”

3. Teksta lauks “Parole (atkārtoti)”

4. Izvēle “Skolēns/Skolotājs”

5. Poga “Ielogoties”

6. Poga “Reģistrēties”

2.4. Lietotāja saskarne “Lietotāja ielogošanas saskarne”. Saskarnes struktūra: skat. 1.7. att.

Lietotājvārds:

Parole:

Reģistrēties

Ielogoties

Iziet

att. 1.7. Lietotāja ielogošanas saskarne

Saskarnes elementi:

1. teksta lauks “Lietotājvārds”

2. teksta lauks ‘Parole”

3. poga “Iziet”

4. poga “Ielogoties”

5. poga “Reģistrēties”

2.5. Lietotāja saskarne “Lietotāja sākumlapa”. Saskarnes struktūra: skat. 1.8. att.

**Atteikties no konta**

Mani testi

Pievienot foto

**Vārds:**

**Uzvārds:**

**E-pasts:**

Saglabāt

Dzēst

**Lietotājvārds:**

**Parole:**

att. 1.8. Lietotāja sākumlapas saskarne

Saskarnes elementi:

1. Poga „Mani testi”

2. Poga „Atteikties no konta”

3. Poga „Pievienot foto”

4. Teksta lauks „Vārds”

5. Teksta lauks „Uzvārds”

6. Teksta lauks „E-pasts”

7. Poga „Dzēst”

8. Poga „Saglabāt”

9. Teksta virkne “Lietotājvārds”

10. Teksta virkne “Parole”

2.6. Lietotāja saskarne “Testu saraksts”. Saskarnes struktūra: skat. 1.9. att.

**Atteikties no konta**

Mans profils

**Meklēšana**

1. Tests

2. Tests

3. Tests

4. Tests

5. Tests

Pildīt testu

Testu statuss

Izveidot testu

Rediģēt testu

Sākt testu

att. 1.9. Testu saraksta saskarne

Saskarnes elementi:

1. Poga „Mans profils”

2. Poga „Atteikties no konta”

3. Teksta lauks „Meklēšana”

4. Testu saraksts

5. Pats tests (viņa nosaukums)

6. Poga „Pildīt testu”

7. Poga „Testu statuss”

8. Poga „Izveidot testu”

9. Poga “Rediģēt testu”

10. Poga “Sākt testu”

2.6. Lietotāja saskarne “Jautājumu logs”. Saskarnes struktūra: skat. 1.10. att.

**Testa nosaukums**

Nākamā lapa

 Jautājuma apraksts

Iepriekšējā lapa

Atbilde

Atbilde

Atbilde

Atbilde

att. 1.10. Jautājumu loga saskarne

Saskarnes elementi:

1. Teksta virkne “Testa nosaukums”

2. Teksta virkne “Jautājuma apraksts”

3. Atbilžu variācijas

4. Poga “Iepriekšējā lapa”

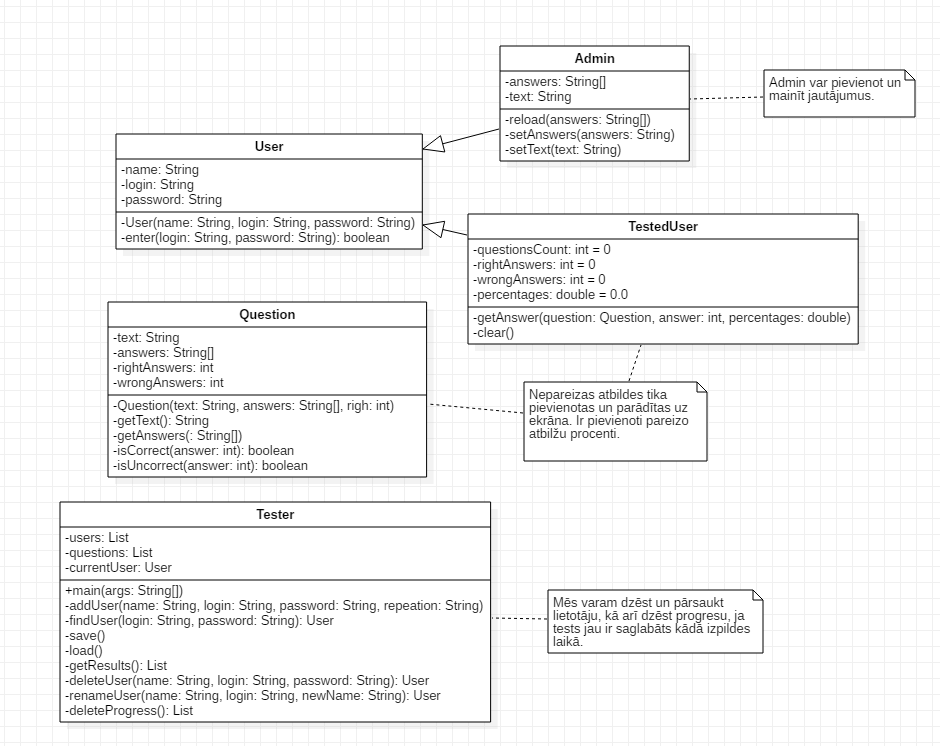
5. Poga “Nākamā lapa”

# Priekšmetiskās jomas klašu sistēmas izstrāde

## Klašu diagramma

Klašu diagrammā ir realizēts vispārīgs klašu apraksts – klašu hierarhijas vispārējā struktūru, to

sadarbība, atribūti, metodes, saskarnes un to savstarpējās attiecības, izmantojot UML valodu.



att. 2.1. Testēšanas sistēmas klašu diagramma UML valodā

Balstoties uz izstrādāto sistēmas modeli var veikt sekojošu klašu iedalījumu (2.1. att.):

1. Klase **User** ir galvenā klase darbam ar lietotājiem.

* User (int id, String login, String password, String firstName, String lastName, String eMail, boolean rights, int[] pointlist) – konstruktors, kurš izveido jaunu lietotāju ar norādītiem datiem. Svarīgākie autorizācijas dati ir lietotājvārds un parole. Pārējos datus var ievadīt pats, kā arī dzēst;
* startTest () – metode, kas ļauj lietotājam veikt testu;
* openQuestion (int questionID) – metode, kas ļauj lietotājam atvērt testā uzdotos jautājumus;
* showStatus () – metode, kas ļauj saskaitīt lietotāja rezultātus;
* nextQuestion () – metode, kas ļauj lietotājam pāriet uz jaunu jautājumu testa darbības laikā;
* previousQuestion() – metode, kas ļauj lietotājam pāriet uz iepriekšējo jautājumu testa darbības laikā;
* getPointlist() – metode, kas atgriež lietotāja punktus testam;
* setTestList () – metode, kas ļauj inicializēt testu;
* getId () – metode, kas atgriež lietotāja id;
* getLogin () – metode, kas atgriež lietotāja lietotājvārdu;
* getPassword () – metode, kas atgriež lietotāja paroli;
* getFirstName () – metode, kas atgriež lietotāja vārdu;
* getLastName () – metode, kas atgriež lietotāja uzvārdu;
* getEmail () – metode, kas atgriež lietotāja pastu;
* getRights () – metode, kas atgriež lietotāja tiesības;
* getTestList () – metode, kas atgriež testu sarakstu;
* openRedacor () – metode, kas ļauj lietotājam ar atļaujām atvērt testa redaktoru;
* changeActivation () – metode, kas ļauj lietotājam ar tiesībām atļaut testa izpildi;
* addTest () – metode, kas ļauj lietotājam ar tiesībām pievienot testu;
* saveQuestionCustomizacion () – metode, kas ļauj lietotājam ar atļaujām saglabāt izmaiņas jautājumos;
* deleteQuestionCustomization () – metode, kas ļauj lietotājam ar atļaujām dzēst izmaiņas jautājumos;
* deleteTest () – metode, kas ļauj lietotājam ar atļaujām dzēst testu.

2. Klase **Teacher** – nodrošina darbu ar lietotajiem un testiem.

* Teacher (int id, String login, String password, String firstName, String lastName, String email, boolean rights, int[] pointlist)– konstruktors, kurš izveido jaunu skolotāju ar norādītiem datiem;
* Klase **Teacher** manto lielāko daļu klases **User** metožu, izņemot datu atgriešanas metodes.

3. Klase **Student** – nodrošina audzēkņa testa pildīšanas darbus.

* Student (int id, String login, String password, String firstName, String lastName, String email, boolean rights, int[] pointlist) – konstruktors, kurš izveido jaunu skolēnu ar norādītiem datiem;
* Student klase nav pārmantojusi daudzas klases User metodes, kā arī nav pārmantojusi nevienu datu atgriešanas metodi. Tas ir tāpēc, ka studentam ir tikai iespēja palaist testu un redzēt rezultātus.

4. Klase **Question** – ir atbildīga par darbu ar testa jautājumiem, to izvadi un ievadītas atbildes

pārbaudi.

* Question (int id, String text, String[] answers, int rightAnswers) – konstruktors, kurš izveido jaunu jautājumu ar norādītiem datiem;
* getId () – metode, kas atgriež jautājumu id;
* getText () – metode, kas atgriež jautājumu saturu;
* getAnswers () – metode, kas atgriež atbilžu variantus;
* getRightAnswers () – metode, kas atgriež pareizās atbildes.

5. Klase **Test** – ir atbildīga par testu izveidi un dzēšanu.

* Test (int id, String name, int[] questionidlist, boolean active) – konstruktors, kurš izveido jaunu testu ar norādītiem datiem;
* isActive() – metode, kas atgriež testa statusu;
* getId() – metode, kas atgriež testa id;
* getName() – metode, kas atgriež testa nosaukumu;
* getQuestionidlist() – metode, kas atgriež testu jautājumu id;
* getQuestionList() – metode, kas atgriež jautājumu sarakstu.

## Klašu realizēšana Java valodā

Izstrādātās klašu diagrammas (2.1. att.) realizēšanai ***NetBeans*** vidē tika izveidots projekts ar

nosaukumu ***TestingSystem***, kurā izstrādātās klases tika aprakstītas ***Java*** valodā. Šeit tiek piedāvāts klašu vispārīgā apraksta (klašu deklarācijas) programmas kods un metožu apraksti.

Pilns avota programmas kods ar klašu metožu definīcijām ir ievietots pielikumā “Pielikums A.

Klašu sistēma”, bet programmas kods, kas realizē šīs klašu sistēmas testēšanu (klase ar grafisko lietotāja saskarni) ir ievietots pielikumā “Pielikums B. testa programmas kods”.public class User {

public Test test;

private int id;

private String login, password, firstName, lastName, eMail;

boolean rights; // "skolotājs" vai "skolēns" (true/false)

int[] pointlist; //punkti par izpildītiem testiem

private Test[] testList;

public User(int id, String login, String password, String firstName, String lastName, String eMail, boolean rights, int[] pointlist) {

this.id = id;

this.login = login;

this.password = password;

this.firstName = firstName;

this.lastName = lastName;

this.eMail = eMail;

this.rights = rights;

this.pointlist = pointlist;

setTestList();

}

public void startTest() {

if (!jframe.jList1.isSelectionEmpty()) {

for (int i = 0; i < jframe.user.getTestList().length; i++) {

if (jframe.jList1.getSelectedValue().equals(jframe.user.getTestList()[i].getName())) {

test = jframe.user.getTestList()[i];

}

}

jframe.radioMemory = new boolean[test.getQuestionList().length][4];

jframe.questionID = 0;

openQuestion(jframe.questionID);

jframe.QuestionPattern.setBounds(0, 0, 420, 339);

jframe.QuestionPattern.setLocationRelativeTo(null);

jframe.QuestionPattern.setVisible(true);

}

}

private void openQuestion(int questionID) {

jframe.buttonGroup2.clearSelection();

jframe.jLabel36.setText(test.getName());

jframe.jLabel37.setText(test.getQuestionList()[questionID].getText());

jframe.jRadioButton1.setText(test.getQuestionList()[questionID].getAnswers()[0]);

jframe.jRadioButton2.setText(test.getQuestionList()[questionID].getAnswers()[1]);

jframe.jRadioButton3.setText(test.getQuestionList()[questionID].getAnswers()[2]);

jframe.jRadioButton4.setText(test.getQuestionList()[questionID].getAnswers()[3]);

jframe.jRadioButton1.setSelected(jframe.radioMemory[questionID][0]);

jframe.jRadioButton2.setSelected(jframe.radioMemory[questionID][1]);

jframe.jRadioButton3.setSelected(jframe.radioMemory[questionID][2]);

jframe.jRadioButton4.setSelected(jframe.radioMemory[questionID][3]);

}

public void showStatus() {

if (!jframe.jList1.isSelectionEmpty()) {

for (int i = 0; i < jframe.user.getTestList().length; i++) {

if (jframe.jList1.getSelectedValue().equals(jframe.user.getTestList()[i].getName())) {

jframe.Stat.setBounds(0, 0, 370, 238);

jframe.Stat.setLocationRelativeTo(null);

jframe.Stat.setVisible(true);

jframe.jLabel21.setText(jframe.user.getTestList()[i].getId() + "");

jframe.jLabel22.setText(jframe.user.getTestList()[i].getQuestionidlist().length + "");

jframe.jLabel25.setText(jframe.user.getTestList()[i].getQuestionidlist().length + "");

}

}

}

}

public void nextQuestion() {

if (jframe.questionID + 1 < test.getQuestionList().length && jframe.questionID >= 0) {

jframe.radioMemory[jframe.questionID] = new boolean[]{jframe.jRadioButton1.isSelected(), jframe.jRadioButton2.isSelected(), jframe.jRadioButton3.isSelected(), jframe.jRadioButton4.isSelected()};

jframe.questionID++;

openQuestion(jframe.questionID);

} else {

jframe.radioMemory[jframe.questionID] = new boolean[]{jframe.jRadioButton1.isSelected(), jframe.jRadioButton2.isSelected(), jframe.jRadioButton3.isSelected(), jframe.jRadioButton4.isSelected()};

int pointsum = 0;

for (int i = 0; i < jframe.radioMemory.length; i++) {

if (jframe.radioMemory[i][test.getQuestionList()[i].getRightAnswers()]) {

pointsum++;

}

}

jframe.jLabel21.setText(test.getId() + "");

jframe.jLabel22.setText(test.getQuestionList().length + "");

jframe.jLabel23.setText(pointsum + "");

jframe.jLabel25.setText(test.getQuestionList().length + "");

double procent = ((double) (pointsum) / (double) (test.getQuestionList().length)) \* 100;

jframe.jLabel26.setText(procent + "");

if (procent < 10) {

jframe.jLabel27.setText("F");

} else if (10 <= procent && procent < 20) {

jframe.jLabel27.setText("E");

} else if (20 <= procent && procent < 30) {

jframe.jLabel27.setText("D");

} else if (30 <= procent && procent < 40) {

jframe.jLabel27.setText("D+");

} else if (40 <= procent && procent < 50) {

jframe.jLabel27.setText("C-");

} else if (50 <= procent && procent < 60) {

jframe.jLabel27.setText("C");

} else if (60 <= procent && procent < 70) {

jframe.jLabel27.setText("B-");

} else if (70 <= procent && procent < 80) {

jframe.jLabel27.setText("B");

} else if (80 <= procent && procent < 90) {

jframe.jLabel27.setText("A-");

} else if (90 <= procent) {

jframe.jLabel27.setText("A");

}

jframe.Stat.setBounds(0, 0, 370, 238);

jframe.Stat.setLocationRelativeTo(null);

jframe.Stat.setVisible(true);

jframe.QuestionPattern.dispose();

}

}

public void previousQuestion() {

if (jframe.questionID < test.getQuestionList().length && jframe.questionID - 1 >= 0) {

jframe.radioMemory[jframe.questionID] = new boolean[]{jframe.jRadioButton1.isSelected(), jframe.jRadioButton2.isSelected(), jframe.jRadioButton3.isSelected(), jframe.jRadioButton4.isSelected()};

jframe.questionID--;

openQuestion(jframe.questionID);

} else {

jframe.Log5.setBounds(0, 0, 464, 125);

jframe.Log5.setLocationRelativeTo(null);

jframe.Log5.setVisible(true);

jframe.TestList.setBounds(0, 0, 620, 296);

jframe.TestList.setLocationRelativeTo(null);

jframe.TestList.setVisible(true);

jframe.QuestionPattern.dispose();

}

}

public int[] getPointlist() {

return pointlist;

}

public void setTestList() {

this.testList = DB\_Test.getTestList().toArray(new Test[0]);

}

public int getId() {

return this.id;

}

public String getLogin() {

return login;

}

public String getPassword() {

return password;

}

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

public String getEmail() {

return eMail;

}

public boolean getRights() {

return rights;

}

public Test[] getTestList() {

setTestList();

return testList;

}

public void openRedacor() {

}

public void changeActivation() {

}

public void addTest() {

}

public void saveQuestionCustomizacion() {

}

public void deleteQuestionCustomization() {

}

public void deleteTest() {

}

}

public class Teacher extends User {

public Test test;

public Teacher(int id, String login, String password, String firstName, String lastName, String email, boolean rights, int[] pointlist) {

super(id, login, password, firstName, lastName, email, rights, pointlist);

}

@Override

public void startTest() {

if (!jframe.jList1.isSelectionEmpty()) {

for (int i = 0; i < jframe.user.getTestList().length; i++) {

if (jframe.jList1.getSelectedValue().equals(jframe.user.getTestList()[i].getName())) {

test = jframe.user.getTestList()[i];

System.out.println(test + " " + jframe.user.getTestList()[i]);

}

}

jframe.radioMemory = new boolean[test.getQuestionList().length][4];

jframe.questionID = 0;

openQuestion(jframe.questionID);

jframe.QuestionPattern.setBounds(0, 0, 420, 339);

jframe.QuestionPattern.setLocationRelativeTo(null);

jframe.QuestionPattern.setVisible(true);

}

}

private void openQuestion(int questionID) {

jframe.buttonGroup2.clearSelection();

jframe.jLabel36.setText(test.getName());

jframe.jLabel37.setText(test.getQuestionList()[questionID].getText());

jframe.jRadioButton1.setText(test.getQuestionList()[questionID].getAnswers()[0]);

jframe.jRadioButton2.setText(test.getQuestionList()[questionID].getAnswers()[1]);

jframe.jRadioButton3.setText(test.getQuestionList()[questionID].getAnswers()[2]);

jframe.jRadioButton4.setText(test.getQuestionList()[questionID].getAnswers()[3]);

jframe.jRadioButton1.setSelected(jframe.radioMemory[questionID][0]);

jframe.jRadioButton2.setSelected(jframe.radioMemory[questionID][1]);

jframe.jRadioButton3.setSelected(jframe.radioMemory[questionID][2]);

jframe.jRadioButton4.setSelected(jframe.radioMemory[questionID][3]);

}

@Override

public void showStatus() {

if (!jframe.jList1.isSelectionEmpty()) {

for (int i = 0; i < jframe.user.getTestList().length; i++) {

if (jframe.jList1.getSelectedValue().equals(jframe.user.getTestList()[i].getName())) {

jframe.Stat.setBounds(0, 0, 370, 238);

jframe.Stat.setLocationRelativeTo(null);

jframe.Stat.setVisible(true);

jframe.jLabel21.setText(jframe.user.getTestList()[i].getId() + "");

jframe.jLabel22.setText(jframe.user.getTestList()[i].getQuestionidlist().length + "");

jframe.jLabel25.setText(jframe.user.getTestList()[i].getQuestionidlist().length + "");

}

}

}

}

@Override

public void openRedacor() {

if (!jframe.jList1.isSelectionEmpty()) {

jframe.jTextArea1.setText("");

jframe.jTextField4.setText("");

jframe.jTextField5.setText("");

jframe.jTextField6.setText("");

jframe.jTextField7.setText("");

jframe.CreateQuestions.setBounds(0, 0, 500, 482);

jframe.CreateQuestions.setLocationRelativeTo(null);

jframe.CreateQuestions.setVisible(true);

for (int i = 0; i < jframe.user.getTestList().length; i++) {

if (jframe.jList1.getSelectedValue().equals(jframe.user.getTestList()[i].getName())) {

test = jframe.user.getTestList()[i];

}

}

DefaultListModel model = new DefaultListModel();

for (int i = 1; i <= test.getQuestionList().length; i++) {

model.addElement("Question " + i);

}

jframe.jList3.setModel(model);

}

}

// Kad skolotājs nodrošina piekļuvi testiem

@Override

public void changeActivation() {

if (!jframe.jList1.isSelectionEmpty()) {

for (int i = 0; i < jframe.user.getTestList().length; i++) {

if (jframe.jList1.getSelectedValue().equals(jframe.user.getTestList()[i].getName())) {

test = jframe.user.getTestList()[i];

}

}

}

if (test.isActive()) {

DB\_Test.activate(test.getId(), false);

} else {

DB\_Test.activate(test.getId(), true);

}

}

@Override

public void addTest() {

if (!jframe.jTextField2.getText().equals("")) {

if ((int) (jframe.jSpinner1.getValue()) > 0) {

int[] questionidlist = new int[(int) (jframe.jSpinner1.getValue())];

for (int i = 0; i < (int) (jframe.jSpinner1.getValue()); i++) {

questionidlist[i] = DB\_Question.addQuestion("question text", new String[]{"answer1", "answer2", "answer3", "answer4"}, 1).getId();

}

DB\_Test.addTest(jframe.jTextField2.getText(), questionidlist);

DefaultListModel model = new DefaultListModel();

for (Test test : jframe.user.getTestList()) {

if (test.isActive() || jframe.user.getRights()) {

model.addElement(test.getName());

}

}

jframe.jList1.setModel(model);

jframe.CreateTest.setVisible(false);

}

}

}

@Override

public void saveQuestionCustomizacion() {

Question question = null;

for (int i = 0; i < jframe.user.getTestList().length; i++) {

if (jframe.jList1.getSelectedValue().equals(jframe.user.getTestList()[i].getName())) {

question = jframe.user.getTestList()[i].getQuestionList()[jframe.jList3.getSelectedIndex()];

}

}

String[] answers = {jframe.jTextField4.getText(), jframe.jTextField5.getText(), jframe.jTextField6.getText(), jframe.jTextField7.getText()};

int rightanswer = 0;

boolean[] buttons = {jframe.jRadioButton5.isSelected(), jframe.jRadioButton6.isSelected(), jframe.jRadioButton7.isSelected(), jframe.jRadioButton8.isSelected()};

for (int i = 0; i < buttons.length; i++) {

if (buttons[i]) {

rightanswer = i;

}

}

DB\_Question.update(question.getId(), jframe.jTextArea1.getText(), answers, rightanswer);

}

// Izdzēst rindu saturu, rediģējot testu

@Override

public void deleteQuestionCustomization() {

jframe.jTextArea1.setText("");

jframe.jTextField4.setText("");

jframe.jTextField5.setText("");

jframe.jTextField6.setText("");

jframe.jTextField7.setText("");

}

@Override

public void nextQuestion() {

if (jframe.questionID + 1 < test.getQuestionList().length && jframe.questionID >= 0) {

jframe.radioMemory[jframe.questionID] = new boolean[]{jframe.jRadioButton1.isSelected(), jframe.jRadioButton2.isSelected(), jframe.jRadioButton3.isSelected(), jframe.jRadioButton4.isSelected()};

jframe.questionID++;

openQuestion(jframe.questionID);

} else {

jframe.radioMemory[jframe.questionID] = new boolean[]{jframe.jRadioButton1.isSelected(), jframe.jRadioButton2.isSelected(), jframe.jRadioButton3.isSelected(), jframe.jRadioButton4.isSelected()};

int pointsum = 0;

for (int i = 0; i < jframe.radioMemory.length; i++) {

if (jframe.radioMemory[i][test.getQuestionList()[i].getRightAnswers()]) {

pointsum++;

}

}

jframe.jLabel21.setText(test.getId() + "");

jframe.jLabel22.setText(test.getQuestionList().length + "");

jframe.jLabel23.setText(pointsum + "");

jframe.jLabel25.setText(test.getQuestionList().length + "");

double procent = ((double) (pointsum) / (double) (test.getQuestionList().length)) \* 100;

jframe.jLabel26.setText(procent + "%");

if (procent < 10) {

jframe.jLabel27.setText("F");

} else if (10 <= procent && procent < 20) {

jframe.jLabel27.setText("E");

} else if (20 <= procent && procent < 30) {

jframe.jLabel27.setText("D");

} else if (30 <= procent && procent < 40) {

jframe.jLabel27.setText("D+");

} else if (40 <= procent && procent < 50) {

jframe.jLabel27.setText("C-");

} else if (50 <= procent && procent < 60) {

jframe.jLabel27.setText("C");

} else if (60 <= procent && procent < 70) {

jframe.jLabel27.setText("B-");

} else if (70 <= procent && procent < 80) {

jframe.jLabel27.setText("B");

} else if (80 <= procent && procent < 90) {

jframe.jLabel27.setText("A-");

} else if (90 <= procent) {

jframe.jLabel27.setText("A");

}

jframe.Stat.setBounds(0, 0, 370, 238);

jframe.Stat.setLocationRelativeTo(null);

jframe.Stat.setVisible(true);

jframe.QuestionPattern.dispose();

}

}

@Override

public void previousQuestion() {

if (jframe.questionID < test.getQuestionList().length && jframe.questionID - 1 >= 0) {

jframe.radioMemory[jframe.questionID] = new boolean[]{jframe.jRadioButton1.isSelected(), jframe.jRadioButton2.isSelected(), jframe.jRadioButton3.isSelected(), jframe.jRadioButton4.isSelected()};

jframe.questionID--;

openQuestion(jframe.questionID);

} else {

jframe.TestList.setBounds(0, 0, 620, 296);

jframe.TestList.setLocationRelativeTo(null);

jframe.TestList.setVisible(true);

jframe.QuestionPattern.dispose();

}

}

@Override

public void deleteTest() {

DB\_Test.delete(test.getId());

}

}

public class Student extends User {

public Test test;

public Student(int id, String login, String password, String firstName, String lastName, String email, boolean rights, int[] pointlist) {

super(id, login, password, firstName, lastName, email, rights, pointlist);

}

@Override

public void startTest() {

if (!jframe.jList1.isSelectionEmpty()) {

for (int i = 0; i < jframe.user.getTestList().length; i++) {

if (jframe.jList1.getSelectedValue().equals(jframe.user.getTestList()[i].getName())) {

test = jframe.user.getTestList()[i];

System.out.println(jframe.user.getTestList()[i].getName());

}

}

jframe.radioMemory = new boolean[test.getQuestionList().length][4];

jframe.questionID = 0;

openQuestion(jframe.questionID);

jframe.QuestionPattern.setBounds(0, 0, 420, 339);

jframe.QuestionPattern.setLocationRelativeTo(null);

jframe.QuestionPattern.setVisible(true);

}

}

// Kad skolotājs ir pievienojis testu, mums ir jāzina jautājumu un atbilžu saturs

private void openQuestion(int questionID) {

jframe.buttonGroup2.clearSelection();

jframe.jLabel36.setText(test.getName());

jframe.jLabel37.setText(test.getQuestionList()[questionID].getText());

jframe.jRadioButton1.setText(test.getQuestionList()[questionID].getAnswers()[0]);

jframe.jRadioButton2.setText(test.getQuestionList()[questionID].getAnswers()[1]);

jframe.jRadioButton3.setText(test.getQuestionList()[questionID].getAnswers()[2]);

jframe.jRadioButton4.setText(test.getQuestionList()[questionID].getAnswers()[3]);

jframe.jRadioButton1.setSelected(jframe.radioMemory[questionID][0]);

jframe.jRadioButton2.setSelected(jframe.radioMemory[questionID][1]);

jframe.jRadioButton3.setSelected(jframe.radioMemory[questionID][2]);

jframe.jRadioButton4.setSelected(jframe.radioMemory[questionID][3]);

}

// Testa rezultātu rādīšana

@Override

public void showStatus() {

if (!jframe.jList1.isSelectionEmpty()) {

for (int i = 0; i < jframe.user.getTestList().length; i++) {

if (jframe.jList1.getSelectedValue().equals(jframe.user.getTestList()[i].getName())) {

jframe.Stat.setBounds(0, 0, 370, 238);

jframe.Stat.setLocationRelativeTo(null);

jframe.Stat.setVisible(true);

jframe.jLabel21.setText(jframe.user.getTestList()[i].getId() + "");

jframe.jLabel22.setText(jframe.user.getTestList()[i].getQuestionidlist().length + "");

jframe.jLabel25.setText(jframe.user.getTestList()[i].getQuestionidlist().length + "");

}

}

}

}

// Metode pārejai uz nākamo jautājumu un rezultātu aprēķināšanai

@Override

public void nextQuestion() {

if (jframe.questionID + 1 < test.getQuestionList().length && jframe.questionID >= 0) {

jframe.radioMemory[jframe.questionID] = new boolean[]{jframe.jRadioButton1.isSelected(), jframe.jRadioButton2.isSelected(), jframe.jRadioButton3.isSelected(), jframe.jRadioButton4.isSelected()};

jframe.questionID++;

openQuestion(jframe.questionID);

} else {

jframe.radioMemory[jframe.questionID] = new boolean[]{jframe.jRadioButton1.isSelected(), jframe.jRadioButton2.isSelected(), jframe.jRadioButton3.isSelected(), jframe.jRadioButton4.isSelected()};

int pointsum = 0;

for (int i = 0; i < jframe.radioMemory.length; i++) {

if (jframe.radioMemory[i][test.getQuestionList()[i].getRightAnswers()]) {

pointsum++;

}

}

jframe.jLabel21.setText(test.getId() + "");

jframe.jLabel22.setText(test.getQuestionList().length + "");

jframe.jLabel23.setText(pointsum + "");

jframe.jLabel25.setText(test.getQuestionList().length + "");

double procent = ((double) (pointsum) / (double) (test.getQuestionList().length)) \* 100;

jframe.jLabel26.setText(procent + "");

if (procent < 10) {

jframe.jLabel27.setText("F");

} else if (10 <= procent && procent < 20) {

jframe.jLabel27.setText("E");

} else if (20 <= procent && procent < 30) {

jframe.jLabel27.setText("D");

} else if (30 <= procent && procent < 40) {

jframe.jLabel27.setText("D+");

} else if (40 <= procent && procent < 50) {

jframe.jLabel27.setText("C-");

} else if (50 <= procent && procent < 60) {

jframe.jLabel27.setText("C");

} else if (60 <= procent && procent < 70) {

jframe.jLabel27.setText("B-");

} else if (70 <= procent && procent < 80) {

jframe.jLabel27.setText("B");

} else if (80 <= procent && procent < 90) {

jframe.jLabel27.setText("A-");

} else if (90 <= procent) {

jframe.jLabel27.setText("A");

}

jframe.Stat.setBounds(0, 0, 370, 238);

jframe.Stat.setLocationRelativeTo(null);

jframe.Stat.setVisible(true);

jframe.QuestionPattern.dispose();

}

}

// Metode pārejai uz iepriekšējo jautājumu

@Override

public void previousQuestion() {

if (jframe.questionID < test.getQuestionList().length && jframe.questionID - 1 >= 0) {

jframe.radioMemory[jframe.questionID] = new boolean[]{jframe.jRadioButton1.isSelected(), jframe.jRadioButton2.isSelected(), jframe.jRadioButton3.isSelected(), jframe.jRadioButton4.isSelected()};

jframe.questionID--;

openQuestion(jframe.questionID);

} else {

jframe.TestList.setBounds(0, 0, 620, 296);

jframe.TestList.setLocationRelativeTo(null);

jframe.TestList.setVisible(true);

jframe.QuestionPattern.dispose();

}

}

private void setRights(boolean rights) {

this.rights = rights;

}

}

# Testa programmas izstrāde

## Testēšanas metodikas

Pirms priekšmetiskās jomas klašu sistēmas testēšanas programmas izstrādes, ir jānosaka testēšanas metodika.

Ir jāatšķir koda testēšana un atkļūdošana. Atkļūdošanu veic programmētājs, izmantojot iebūvētos izstrādes vides rīkus un balstoties uz pieredzi programmas koda rakstīšanā. Būtībā runa iet par sintaktisko un semantisko kļūdu identificēšanu programmas tekstā.

Testēšana ir process, kurā nepieciešams plānot un īstenot vairākas sākotnējās procedūras, no kurām galvenā ir testa piemēru kopas izstrāde, kas veido testa plānu. Testpiemēri vairumā gadījumu ir balstīti uz sistēmas funkcionālajām prasībām un var ietekmēt dažādus attīstības līmeņus (vienības testēšana, integrācijas testēšana, sistēmas testēšana).

Projekta darba laikā nepieciešams veikt vienkāršotu vienības testēšanas versiju, kas realizē visu izstrādāto klašu metožu testēšanu. Ar testēšanas metodiku projekta darbā pieņem dažādu klases metožu izsaukšanas iespēju sarakstu ar gaidāmajiem rezultātiem un šo testēšanas izsaukumu izpildes kārtību.

## Testēšanas programmas struktūra

Testēšanas programmas projektēšana un programmatūras izstrāde ir ļoti līdzīga iepriekš aprakstītajam klašu sistēmas izstrādes procesam.

Jāatzīmē tie momenti un prasības, kas pastāv, izstrādājot testēšanas programmu.

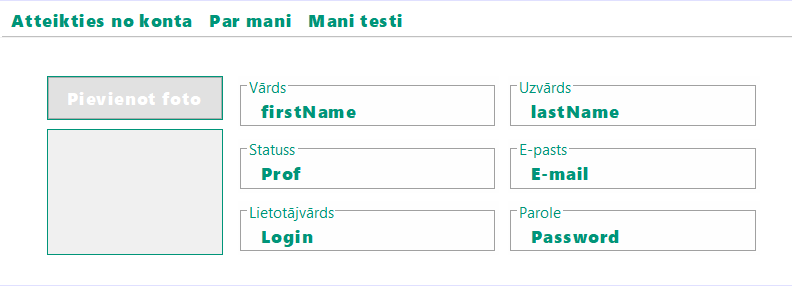
* Testēšanas programmai ir loga interfeiss, kas ietver sevī izvēlnes un interaktīvā interfeisa rīkus, ar kuru palīdzību var pārbaudīt izveidoto klašu sistēmu.
* Interfeisa elementu (pogas, saraksti, ievades lauki, dialoglodziņi, grafiskie elementi utt.) komplekts tiek noteikts, pamatojoties uz iepriekšējā sadaļā aprakstīto testēšanas metodiku.
* Testēšanas programmas interfeisa kvalitātei jābūt vienkāršai lietošanā un interaktīvai. Galvenā prasība ir spēja realizēt testēšanas metodes.

1. Ielogošanas procesa apraksts:

* Ielogošanas process paredzēts reģistrēto lietotāju pievienošanai sistēmā. Lai lietotājs varētu piekļūt testam viņam pēc ir nepieciešams autorizēties. Ielogošanas procesā lietotājs ievada iepriekš reģistrētos datus – lietotājvārdu un paroli (3.1. att.). Ja lietotāja dati sakrīt ar validācijas datiem, tad lietotājam tiek atvērts programmas galvenais logs (3.2. att.).
* Ielogošanas procesā ir iespējamas vairākas kļūdas – neaizpildīti lauki, neprecīzi ievadīti lietotāja dati.



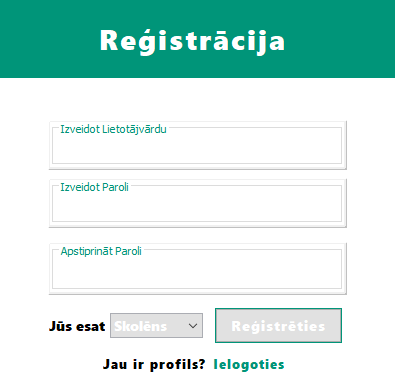
att. 3.1. Testēšanas sistēmas ielogošanas forma



att. 3.2. Programmas galvenais logs

2. Reģistrācijas procesa apraksts:

* Reģistrācijas process ir nepieciešams tiem, kuri nav reģistrēti programmas datubāzē. Lai autorizētos sistēmā, ir nepieciešams reģistrēties. Reģistrācijas procesā lietotājam ir jāievada lietotājvārds un parole, kā arī drošības nolūkos jāatkārto parole un jāizvēlas piederība: skolēns vai skolotājs (3.3. att.). Ja reģistrācija ir veiksmīga, parādīsies attiecīgs logs (3.4. att.).
* Reģistrācijas procesā ir iespējamas vairākas kļūdas – neaizpildīti lauki, neprecīzi ievadīti lietotāja dati.



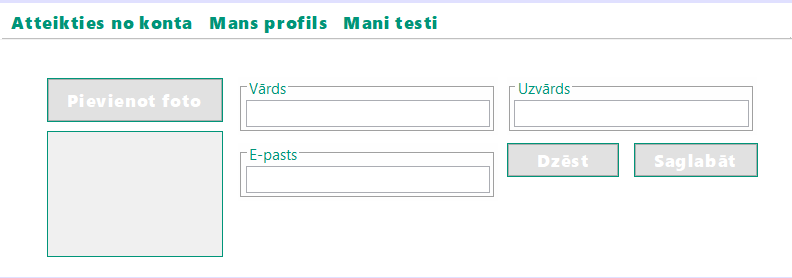
att. 3.3. Testēšanas sistēmas reģistrēšanas forma



att. 3.4. Veiksmīgai reģistrācijai logs

3. Datu aizpildīšanas apraksts.

* Kad esam veiksmīgi autorizējušies (3.1. att.), mēs tiekam izmesti uz profila logu (3.2. att.). Sākotnēji mūsu datu lauki ir tukši, izņemot lietotājvārdu un paroli, taču tos var aizpildīt lietotāja datu aizpildīšanas logā (3.5. att.). No profila loga mēs varam iziet no konta, kā arī pāriet uz testu saraksta logu (3.6. att.) un sākt testu.



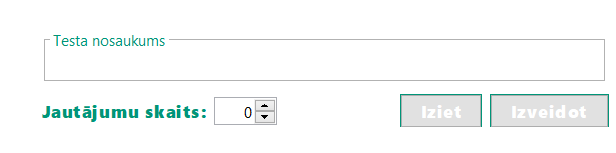
att. 3.5. Lietotāja datu ievadīšanas logs



att. 3.6. Testu saraksta logs un darbs ar tiem

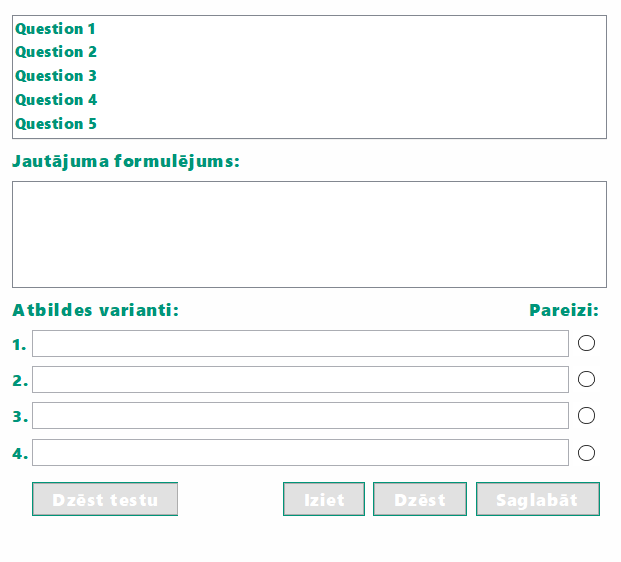
4. Darba ar testiem apraksts.

* Ja esam skolēns, mums būs pieejamas tikai divas pogas: Pildīt un Statuss. Skolotājam ir pieejamas visas iespējamās pogas darbam ar testu izveidi un rediģēšanu (3.6. att.).
* Ja vēlamies izveidot testu, noklikšķiniet uz pogas Izveidot, un parādās attiecīgais izveides logs. Tajā ir norādīts testa nosaukums un jautājumu skaits (3.7. att.).



att. 3.7. Testu veidošanas logs

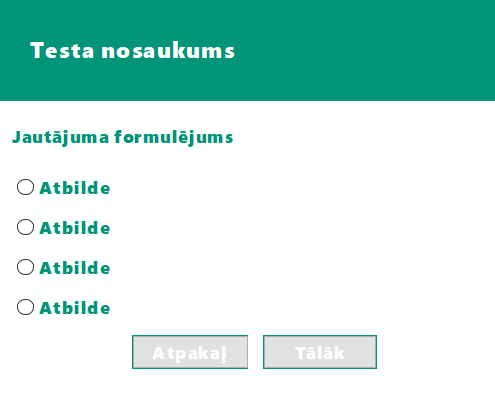
* Pēc tam, kad mūsu tests ir parādījies testu sarakstā un ir atlasīts, mēs varam to rediģēt, nospiežot pogu Rediģēt. Parādīsies logs Rediģēt jautājumus (3.6. att.), kurā mēs izvēlamies jautājumu, ierakstām tā formulējumu, ierakstām atbilžu variantus un to, kurš no tiem būs pareizs (3.8. att.).



att. 3.8. Testu rediģēšanas logs

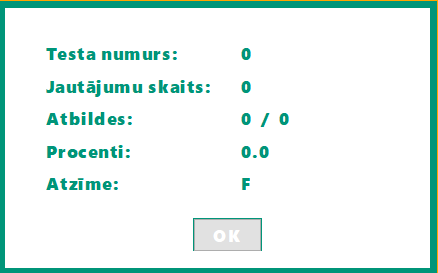
5. Testa izpildes un rezultātu apraksts.

* Kā parasts lietotājs mēs veiksim testus, izvēloties tos no testu saraksta (3.6. att.). Lai izvairītos no loga izveides katram jautājumam, mēs izmantosim šablona logu, kas ir pielāgots testa datiem (3.9. att.).



att. 3.9. Testa šablona logs

* Pēc testa pabeigšanas mums tiks parādīts rezultātu logs ar mūsu rezultātiem (3.10. att.).

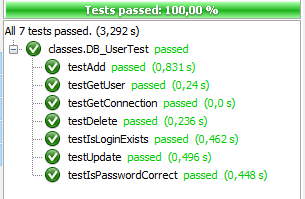


att. 3.10. Rezultātus izvades logs

## Testēšanas rezultāti

1. Klases testa rezultāti darbam ar lietotāju datubāzi.

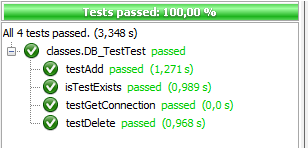
* Testēšanā tika pārbaudītas tādas metodes kā lietotāja pievienošana, lietotāja pārbaude, lietotāja datu maiņa, lietotāja dzēšana, vai pastāv lietotāja pieteikšanās vārds, vai pastāv lietotāja parole (pārbaudei autorizācijas laikā).



att. 3.11. Darba ar lietotāju datubāzi testēšanas rezultāti

1. Klases testa rezultāti ar testa datubāzi.

Tika pārbaudītas tādas testēšanas metodes kā testa pievienošana, testa maiņa un tas, vai tests eksistē.



att. 3.12. Darba ar testu datubāzi testēšanas rezultāti

# Lietotāja rokasgrāmata

## Lietotāja instrukcija

1. Lietotājs palaiž programmu.
2. Piesakās sistēmā, izmantojot savu lietotājvārdu un paroli, ja lietotājam nav sava konta, tad viņš var reģistrēt to.
   1. Reģistrācijas logā lietotājs aizpilda visus nepieciešamus datus un noklikšķina uz pogas „Reģistrēties”.
3. Lietotāja izvēlnē ir 5 pogas.
   1. Poga «Pildīt testu» ļauj sākt pildīt testu.
      1. Pēc testa izpildīšanas tiek paradīts testa izpildes rezultāti.
      2. Pēc testa pabeigšanas un iziešanas no testa ir iespējams vēlreiz apskatīt rezultātus, izmantojot pogu «Statuss».
   2. Poga «Atteikties no konta» ļauj iziet no sava konta.
   3. Poga «Par mani» ļauj ierakstīt un saglabāt savus datus.
   4. Poga «Mani testi» ļauj apskatīt savus testus.
4. Skolotāja izvēlnē ir 7 pogas.
   1. Poga «Izveidot testu» ļauj skolotājam izveidot testu.
   2. Poga «Rediģēt testu» ļauj skolotājam rediģēt testu: testa nosaukums, jautājuma saturs, jautājuma pievienošana, jautājuma dzēšana.
   3. Poga «Sākt testu» ļauj atļaut lietotājam piekļuvi testam.
   4. Poga «Statuss» ļauj paradīt testa statusu.
   5. Poga «Atteikties no konta» ļauj iziet no skolotāja konta.
   6. Poga «Par mani» ļauj ierakstīt un saglabāt savus datus.
   7. Poga «Mani testi» ļauj apskatīt savus testus.

## Palīdzības sistēma (Help)

1. Lai ieietu programmā, ir jāveic autorizācija: ievadiet savu lietotājvārdu un paroli un noklikšķiniet uz pogas Autorizēt.

1.1. Ja jums nav sava konta, izlasiet uzrakstu zem pogām un noklikšķiniet uz Reģistrēties.

1.1.1. Reģistrācijas logā jāraksta šādi dati: lietotājvārds, parole un vēlreiz parole apstiprināšanai un drošībai. Lūdzu, ņemiet vērā, ka lietotājvārdam jābūt unikālam.

1.1.2. Nākamais punkts ir izvēlēties savu piederību: skolēns vai skolotājs. Ja izvēlaties skolēnu, varēsiet veikt tikai testus un apskatīt rezultātus. Ja izvēlaties skolotāju, viņš/viņa varēs ne tikai darīt to, ko var darīt skolēns, bet arī izveidot testus, dzēst testus, rediģēt testā iekļautos jautājumus un atļaut izpildīt testus.

1.1.3. Ja vēlaties atgriezties autorizācijas logā, izlasiet uzrakstu zem pogām un noklikšķiniet uz Ielogoties.

1.2. Ja vēlaties iziet, autorizācijas logā noklikšķiniet uz Iziet.

2. Profila logā tiks norādīti jūsu dati: vārds, uzvārds, amats, piederība, kā arī lietotājvārds un parole, lai lietotājs tos vienmēr varētu atcerēties. Ir iespējams pievienot arī savu profila fotoattēlu.

2.1. Lai aizpildītu savus datus, izmantojiet pogu Par mani, lai pārietu uz datu aizpildīšanas logu, ievadiet savus datus un saglabājiet tos.

2.2. Lai apskatītu pieejamos testus, noklikšķiniet uz pogas Mani testi, lai pārietu uz testu saraksta logu.

2.3. Ja vēlaties iziet no profila, noklikšķiniet uz pogas Atteikties no konta.

3. Testu saraksta logā mūs sagaidīs pieejamo testu saraksts un pieejamās pogas.

3.1. Lai sāktu testu, nospiediet pogu Pildīt.

3.2. Lai apskatītu testa rezultātus, noklikšķiniet uz pogas Statuss.

3.3. Poga Pievienot ļauj pievienot testu, ierakstot tā nosaukumu un izvēloties jautājumu skaitu.

3.3.1. Kad esam pievienojuši testu, mums jāiziet no testa izveides loga, jāizvēlas mūsu jaunais tests un jānospiež poga Rediģēt. Šādā veidā mēs varam rediģēt tā jautājumus un dzēst to.

3.4. Poga Sākt ļauj skolotājam sākt testus skolēniem.

# Secinājumi

Tika izstrādāts projekts «Testēšanas sistēma» NetBeans vidē. Tika izpildītas visas programmas bāzes prasības. Darba gaitā tika izveidota programma, kurā mēs:

* Mēs varam pieteikties;
* Varam reģistrēties;
* Varam pievienot un mainīt savus datus profilā;
* Varam apskatīt pieejamo testu sarakstu;
* Varam izveidot, rediģēt un dzēst testus.
* Varam kārtot pārbaudes darbus un iegūt atzīmes.

Darbā bija daudz kļūdu un grūtību. Manas nolaidības dēļ darbā tika nokavēts darba iesniegšanas termiņš. Taču mums ir iespēja mācīties no savām kļūdām un turpmāk tās nepieļaut.

# Izmantotās literatūras (informācijas avotu) saraksts

1. **Dortiņa, I.** 5.2.1. Versiju kontroles sistēmas. *e.dttt.lv.* [Tiešsaiste] 2024. gada 31. 05. [Citēts: 2024. gada 17. 06.] http://e.dttt.lv/mod/resource/view.php?id=4462.

2. **Dortiņa, I. un Dauģerts, A.** Projekta darbs Nr.1. Testēšanas sistēmas izveidošana. *e.dttt.lv.* [Tiešsaiste] 2024. gada 31. 05. [Citēts: 2024. gada 17. 06.] http://e.dttt.lv/mod/resource/view.php?id=4438.

3. —. Projekta darba paskaidrojošā apraksta paraugs. *e.dttt.lv.* [Tiešsaiste] 2024. gada 31. 05. [Citēts: 2024. gada 17. 06.] http://e.dttt.lv/mod/resource/view.php?id=4437.

4. —. 5.1.4. Automatizētu testu izstrāde. *e.dttt.lv.* [Tiešsaiste] 2024. gada 24. 05. [Citēts: 2024. gada 17. 06.] http://e.dttt.lv/mod/resource/view.php?id=4455.

5. —. 4.2.1. Paralēlie uzdevumi, procesi un pavedieni. *e.dttt.lv.* [Tiešsaiste] 2024. gada 02. 04. [Citēts: 2024. gada 17. 06.] http://e.dttt.lv/mod/resource/view.php?id=4427.

6. —. 4.1.1. Java. Grafiskā lietotāja saskarne. *e.dttt.lv.* [Tiešsaiste] 2024. gada 13. 02. [Citēts: 2024. gada 17. 06.] http://e.dttt.lv/mod/resource/view.php?id=4393.

7. —. 4.1.2. Java GUI izveidošana NetBeans vidē. *e.dttt.lv.* [Tiešsaiste] 2024. gada 20. 02. [Citēts: 2024. gada 17. 06.] http://e.dttt.lv/mod/resource/view.php?id=4407.

8. —. Praktiskais darbs Nr. 16. SDI lietotnes izstrāde ar NetBeans GUI Builder. *e.dttt.lv.* [Tiešsaiste] 2024. gada 20. 02. [Citēts: 2024. gada 17. 06.] http://e.dttt.lv/mod/resource/view.php?id=4417.

9. **How to add an image to a JPanel? *stackoverflow.* [Tiešsaiste] 2008. gada 18. 11. [Citēts: 2024. gada 17. 06.] https://stackoverflow.com/questions/299495/how-to-add-an-image-to-a-jpanel.**

# Pielikums A. Klašu sistēma

public class DB\_User {

private static final String DB\_URL = "jdbc:derby://localhost:1527/DB";

private static final String DB\_USER = "nbuser";

private static final String DB\_PASSWORD = "nbuser";

private static ArrayList<User> userList;

public static Connection getConnection() {

Connection conn = null;

try {

conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);

} catch (SQLException e) {

System.out.println(e);

}

return conn;

}

public static User addUser(String login, String password, boolean rights) {

String query = "INSERT INTO APP.USERS (LOGIN, PASSWORD, RIGHTS) VALUES (?, ?, ?)";

try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query, Statement.RETURN\_GENERATED\_KEYS)) {

pstmt.setString(1, login);

pstmt.setString(2, password);

pstmt.setBoolean(3, rights);

pstmt.executeUpdate();

getUserList();

System.out.println(rights);

return userList.get(userList.size() - 1);

} catch (SQLException e) {

System.out.println(e);

}

return null;

}

public static void updateUser(int id, String firstName, String lastName, String eMail) {

String query = "UPDATE APP.USERS SET FIRSTNAME = ?, LASTNAME = ?, EMAIL = ? WHERE ID = ?";

try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setString(1, firstName);

pstmt.setString(2, lastName);

pstmt.setString(3, eMail);

pstmt.setInt(4, id);

pstmt.executeUpdate();

getUserList();

} catch (SQLException e) {

System.out.println(e);

}

}

public static void delete(int id) {

String query = "DELETE FROM APP.USERS WHERE ID = ?";

try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setInt(1, id);

pstmt.executeUpdate();

userList.removeIf(user -> user.getId() == id);

} catch (SQLException e) {

System.out.println(e);

}

}

public static User getUser(String login, String password) {

getUserList();

for (User user : userList) {

if (user.getLogin().equals(login) && user.getPassword().equals(password)) {

return user;

}

}

return null;

}

public static User getUser(int id) {

getUserList();

for (User user : userList) {

if (user.getId() == id) {

return user;

}

}

return null;

}

public static void getUserList() {

userList = new ArrayList<User>();

String query = "SELECT \* FROM APP.USERS";

try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query); ResultSet rs = pstmt.executeQuery()) {

while (rs.next()) {

int id = rs.getInt(1);

String login = rs.getString(2);

String password = rs.getString(3);

String firstName = rs.getString(4);

String lastName = rs.getString(5);

String email = rs.getString(6);

boolean rights = rs.getBoolean(7);

int[] pointlist = null;

if (rs.getString(8) != null) {

String[] buf = rs.getString(8).split(" ");

pointlist = new int[buf.length];

for (int i = 0; i < buf.length; i++) {

pointlist[i] = Integer.parseInt(buf[i]);

}

}

if (rights) {

userList.add(new Teacher(id, login, password, firstName, lastName, email, rights, pointlist));

} else {

userList.add(new Student(id, login, password, firstName, lastName, email, rights, pointlist));

}

}

} catch (SQLException e) {

System.out.println(e);

}

}

public static boolean isLoginExists(String login) {

getUserList();

for (User user : userList) {

if (user.getLogin().equals(login)) {

return true;

}

}

return false;

}

// Punktu pievienošana par jautājumu aizpildīšanu

public static void addPointList(int id) {

String query = "UPDATE APP.USERS SET POINTLIST = ? WHERE ID = ?";

try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query)) {

String buf = "";

for (int i : getUser(id).getPointlist()) {

buf += i + " ";

}

buf += 0 + " ";

pstmt.setString(1, buf);

pstmt.setInt(2, id);

pstmt.executeUpdate();

getUserList();

} catch (SQLException e) {

System.out.println(e);

}

}

// Punktu skaits par jautājumu aizpildīšanu var atšķirties

public static void updatePointList(int id, int testid, int points) {

String query = "UPDATE APP.USERS SET POINTLIST = ? WHERE ID = ?";

try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query)) {

getUserList();

int[] pointlist = getUser(id).getPointlist();

for (int i = 0; i < userList.size(); i++) {

if (userList.get(i).getId() == id) {

pointlist[i] = points;

}

}

String buf = "";

for (int i = 0; i < pointlist.length; i++) {

buf += pointlist[i] + " ";

}

pstmt.setString(1, buf);

pstmt.setInt(2, id);

pstmt.executeUpdate();

getUserList();

} catch (SQLException e) {

System.out.println(e);

}

}

public static boolean isPasswordCorrect(String login, String password) {

getUserList();

for (User user : userList) {

if (user.getLogin().equals(login) && user.getPassword().equals(password)) {

return true;

}

}

return false;

}

}

public class DB\_Test {

private static final String DB\_URL = "jdbc:derby://localhost:1527/DB";

private static final String DB\_USER = "nbuser";

private static final String DB\_PASSWORD = "nbuser";

private static ArrayList<Test> testList;

public static Connection getConnection() {

Connection conn = null;

try {

conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);

} catch (SQLException e) {

System.out.println(e);

}

return conn;

}

public static void addTest(String name, int[] questionidlist) {

getTestList();

String query = "INSERT INTO APP.TESTS (NAME, QUESTIONIDLIST, ACTIVE)"

+ "VALUES (?, ?, ?)";

try ( Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setString(1, name);

String buf = "";

for (int i = 0; i < questionidlist.length; i++) {

buf += questionidlist[i] + " ";

}

pstmt.setString(2, buf);

pstmt.setBoolean(3, false);

pstmt.executeUpdate();

} catch (SQLException e) {

System.out.println(e);

}

getTestList();

}

// Testa datu mainīšana (kad skolotājs nospiež pogu "Sākt")

public static void update(int id, String name, int[] questionIdList, boolean active) {

try ( Connection conn = getConnection(); Statement stmt = conn.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE)) {

String buf = "";

for (int i = 0; i < questionIdList.length; i++) {

buf += questionIdList[i] + " ";

}

stmt.executeUpdate("UPDATE APP.TESTS SET NAME = '" + name + "', QUESTIONIDLIST = '" + buf + "', ACTIVE = '" + active + "' WHERE id = " + id);

} catch (SQLException e) {

System.out.println(e);

}

getTestList();

}

// Testa veikšanas atļaujas pārbaude

public static void activate(int id, boolean active) {

try ( Connection conn = getConnection(); Statement stmt = conn.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE)) {

stmt.executeUpdate("UPDATE APP.TESTS SET ACTIVE = " + active + " WHERE id = " + id);

getTestList();

} catch (SQLException e) {

System.out.println(e);

}

}

public static void delete(int id) {

String query = "DELETE FROM APP.TESTS WHERE id = ?";

try ( Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setInt(1, id);

pstmt.executeUpdate();

} catch (SQLException e) {

System.out.println(e);

}

getTestList();

}

// Testu pārbaude sarakstā

public static ArrayList<Test> getTestList() {

testList = new ArrayList<Test>();

String query = "SELECT \* FROM APP.TESTS";

try ( Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query); ResultSet rs = pstmt.executeQuery()) {

while (rs.next()) {

int id = rs.getInt(1);

String name = rs.getString(2);

String[] buf = rs.getString(3).split(" ");

int[] questionIDList = new int[buf.length];

for (int i = 0; i < buf.length; i++) {

questionIDList[i] = Integer.parseInt(buf[i]);

}

boolean active = rs.getBoolean(4);

testList.add(new Test(id, name, questionIDList, active));

}

} catch (SQLException e) {

System.out.println(e);

}

return testList;

}

public static boolean isTestExists(int id) {

getTestList();

for (int i = 0; i < testList.size(); i++) {

if (testList.get(i).getId() == id) {

return true;

}

}

return false;

}

}

public class DB\_Question {

private static final String DB\_URL = "jdbc:derby://localhost:1527/DB";

private static final String DB\_USER = "nbuser";

private static final String DB\_PASSWORD = "nbuser";

private static ArrayList<Question> questionList;

public static Connection getConnection() {

Connection conn = null;

try {

conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);

} catch (SQLException e) {

System.out.println(e);

}

return conn;

}

// Jautājuma datu mainīšana

public static void update(int id, String text, String[] answers, int rightanswer) {

try (Connection conn = getConnection(); Statement stmt = conn.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE)) {

String buf = "";

for (int i = 0; i < answers.length; i++) {

buf += answers[i] + ";";

}

stmt.executeUpdate("UPDATE APP.QUESTIONS SET text = '" + text + "', answers = '" + buf + "', rightanswer = " + rightanswer + " WHERE id = " + id);

} catch (SQLException e) {

System.out.println(e);

}

getQuestionList();

}

public static void delete(int id) {

String query = "DELETE FROM APP.QUESTIONS WHERE id = ?";

try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setInt(1, id);

pstmt.executeUpdate();

} catch (SQLException e) {

System.out.println(e);

}

getQuestionList();

}

public static Question addQuestion(String text, String[] answers, int rightAnswers) {

String query = "INSERT INTO APP.QUESTIONS (TEXT, ANSWERS, RIGHTANSWER)"

+ "VALUES (?, ?, ?)";

try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setString(1, text);

String buf = "";

for (int i = 0; i < answers.length; i++) {

buf += answers[i] + ";";

}

pstmt.setString(2, buf);

pstmt.setInt(3, rightAnswers);

pstmt.executeUpdate();

getQuestionList();

return questionList.get(questionList.size() - 1);

} catch (SQLException e) {

System.out.println(e);

}

return null;

}

// Noskaidrosim, kādi jautājumi ir iekļauti sarakstā

public static ArrayList<Question> getQuestionList() {

questionList = new ArrayList<Question>();

String query = "SELECT \* FROM APP.QUESTIONS";

try (Connection conn = getConnection(); PreparedStatement pstmt = conn.prepareStatement(query); ResultSet rs = pstmt.executeQuery()) {

while (rs.next()) {

int id = rs.getInt(1);

String text = rs.getString(2);

String[] answers = rs.getString(3).split(";");

int rightAnswer = rs.getInt(4);

questionList.add(new Question(id, text, answers, rightAnswer));

}

} catch (SQLException e) {

System.out.println(e);

}

return questionList;

}

// Šī metode ir nepieciešama, lai noskaidrotu jautājumu kārtas numurus.

// Id nosaka, kurš jautājums pieder testam.

public static int[] getQuestionIDList(int count) {

getQuestionList();

int lastID;

if (questionList.size() != 0) {

lastID = questionList.get(questionList.size() - 1).getId() + 1;

} else {

lastID = 1;

}

int[] questionIDList = new int[count];

for (int i = 0; i < count; i++) {

questionIDList[i] = lastID + i;

}

return questionIDList;

}

// Konkrēta jautājuma meklēšana pēc id

public static Question getQuestion(int id) {

getQuestionList();

for (int i = 0; i < questionList.size(); i++) {

if (questionList.get(i).getId() == id) {

return questionList.get(i);

}

}

return null;

}

}

# Pielikums B. testa programmas kods

public class PD1\_Rinkevica extends javax.swing.JFrame {

public User user;

public int questionID = 0;

public boolean[][] radioMemory = null;

public PD1\_Rinkevica() {

initComponents();

DB\_User.getUserList();

DB\_Test.getTestList();

DB\_Question.getQuestionList();

}

public void signUp() {

String login = jTextField3.getText();

String password = new String(jPasswordField2.getPassword());

String passwordCheck = new String(jPasswordField3.getPassword());

if (jTextField3.getText().equals("") || new String(jPasswordField2.getPassword()).equals("") || new String(jPasswordField3.getPassword()).equals("")) {

Exception2.setBounds(0, 0, 353, 125);

Exception2.setLocationRelativeTo(null);

Exception2.setVisible(true);

} else if (!DB\_User.isLoginExists(login)) {

if (password.equals(passwordCheck)) {

boolean rights = jComboBox1.getSelectedIndex() == 1;

user = DB\_User.addUser(login, password, rights);

Log1.setBounds(0, 0, 353, 125);

Log1.setLocationRelativeTo(null);

Log1.setVisible(true);

} else {

Exception3.setBounds(0, 0, 353, 125);

Exception3.setLocationRelativeTo(null);

Exception3.setVisible(true);

}

} else {

Exception4.setBounds(0, 0, 353, 125);

Exception4.setLocationRelativeTo(null);

Exception4.setVisible(true);

}

}

public void login() {

String login = writeLogin.getText();

String password = new String(jPasswordField1.getPassword());

if (writeLogin.getText().equals("") || new String(jPasswordField1.getPassword()).equals("")) {

Exception2.setBounds(0, 0, 353, 125);

Exception2.setLocationRelativeTo(null);

Exception2.setVisible(true);

} else if (DB\_User.isLoginExists(login)) {

if (DB\_User.isPasswordCorrect(login, password)) {

user = DB\_User.getUser(login, password);

jLabel9.setText(user.getLogin());

jLabel11.setText(user.getPassword());

jLabel13.setText(user.getFirstName());

jLabel15.setText(user.getLastName());

jLabel16.setText(user.getEmail());

jLabel40.setText((user.getRights()) ? "Skolotājs" : "Skolēns");

Profile.setBounds(0, 0, 650, 290);

Profile.setLocationRelativeTo(null);

Profile.setVisible(true);

dispose();

} else {

Exception1.setBounds(0, 0, 353, 125);

Exception1.setLocationRelativeTo(null);

Exception1.setVisible(true);

}

} else {

Exception1.setBounds(0, 0, 353, 125);

Exception1.setLocationRelativeTo(null);

Exception1.setVisible(true);

}

}

private void SignUpActionPerformed(java.awt.event.ActionEvent evt) {

signUp();

}

private void jLabel7MouseClicked(java.awt.event.MouseEvent evt) {

jframe.setVisible(true);

Signup.dispose();

jTextField3.setText("");

jPasswordField2.setText("");

jPasswordField3.setText("");

}

private void exitActionPerformed(java.awt.event.ActionEvent evt) {

System.exit(0);

}

private void lblSignUpMouseClicked(java.awt.event.MouseEvent evt) {

Signup.setBounds(0, 0, 410, 455);

Signup.setLocationRelativeTo(null);

Signup.setVisible(true);

jComboBox1.removeAllItems();

jComboBox1.addItem("Skolēns");

jComboBox1.addItem("Skolotājs");

dispose();

}

private void loginActionPerformed(java.awt.event.ActionEvent evt) {

login();

}

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {

user.startTest();

}

private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {

user.showStatus();

}

private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {

CreateTest.setBounds(0, 0, 530, 170);

CreateTest.setLocationRelativeTo(null);

CreateTest.setVisible(true);

}

private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {

user.openRedacor();

}

private void jButton13ActionPerformed(java.awt.event.ActionEvent evt) {

user.changeActivation();

Log7.setBounds(0, 0, 353, 125);

Log7.setLocationRelativeTo(null);

Log7.setVisible(true);

}

private void jMenu4MouseClicked(java.awt.event.MouseEvent evt) {

TestList.setBounds(0, 0, 600, 310);

TestList.setLocationRelativeTo(null);

TestList.setVisible(true);

Profile.dispose();

if (user.getRights()) {

jButton11.setVisible(true);

jButton12.setVisible(true);

jButton13.setVisible(true);

} else {

jButton11.setVisible(false);

jButton12.setVisible(false);

jButton13.setVisible(false);

}

DefaultListModel model = new DefaultListModel();

for (Test test : user.getTestList()) {

if (test.isActive() || user.getRights()) {

model.addElement(test.getName());

}

}

jList1.setModel(model);

if (model.isEmpty()) {

Exception5.setBounds(0, 0, 353, 125);

Exception5.setLocationRelativeTo(null);

Exception5.setVisible(true);

}

}

private void jMenu3MouseClicked(java.awt.event.MouseEvent evt) {

jframe.setLocationRelativeTo(null);

jframe.setVisible(true);

Profile.dispose();

writeLogin.setText("");

jPasswordField1.setText("");

jTextField3.setText("");

jPasswordField2.setText("");

jPasswordField3.setText("");

}

private void jMenu2MouseClicked(java.awt.event.MouseEvent evt) {

Profile.setBounds(0, 0, 650, 290);

Profile.setLocationRelativeTo(null);

Profile.setVisible(true);

TestList.dispose();

}

private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {

JFileChooser fileChooser = new JFileChooser();

int returnValue = fileChooser.showOpenDialog(null);

if (returnValue == JFileChooser.APPROVE\_OPTION) {

File selectedFile = fileChooser.getSelectedFile();

try {

BufferedImage img = ImageIO.read(selectedFile);

image = img;

jPanel17.repaint();

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

private void jMenu5MouseClicked(java.awt.event.MouseEvent evt) {

CreateDate.setBounds(0, 0, 650, 255);

CreateDate.setLocationRelativeTo(null);

CreateDate.setVisible(true);

Profile.dispose();

}

private void jMenu6MouseClicked(java.awt.event.MouseEvent evt) {

CreateDate.setBounds(0, 0, 650, 255);

CreateDate.setLocationRelativeTo(null);

CreateDate.setVisible(true);

TestList.dispose();

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

jTextField2.getText();

int spinnerValue = (int) jSpinner1.getValue();

if (jTextField2.equals("") || spinnerValue <= 0) {

Exception2.setBounds(0, 0, 353, 125);

Exception2.setLocationRelativeTo(null);

Exception2.setVisible(true);

} else {

user.addTest();

Log6.setBounds(0, 0, 353, 125);

Log6.setLocationRelativeTo(null);

Log6.setVisible(true);

}

}

private void jList3MouseClicked(java.awt.event.MouseEvent evt) {

try {

Thread.sleep(200);

} catch (InterruptedException e) {

System.out.println(e);

}

Question question = null;

for (int i = 0; i < user.getTestList().length; i++) {

if (jList1.getSelectedValue().equals(user.getTestList()[i].getName())) {

System.out.println(user.getTestList()[i]);

question = user.getTestList()[i].getQuestionList()[jList3.getSelectedIndex()];

}

}

jTextArea1.setText(question.getText());

jTextField4.setText(question.getAnswers()[0]);

jTextField5.setText(question.getAnswers()[1]);

jTextField6.setText(question.getAnswers()[2]);

jTextField7.setText(question.getAnswers()[3]);

JRadioButton[] radioButtonList = {jRadioButton5, jRadioButton6, jRadioButton7, jRadioButton8};

radioButtonList[question.getRightAnswers()].setSelected(true);

}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

if (buttonGroup1.getSelection() == null || jTextField4.equals("") || jTextField5.equals("") || jTextField6.equals("") || jTextField7.equals("") || jTextArea1.equals("")) {

Exception2.setBounds(0, 0, 353, 125);

Exception2.setLocationRelativeTo(null);

Exception2.setVisible(true);

} else {

user.saveQuestionCustomizacion();

Log3.setBounds(0, 0, 353, 125);

Log3.setLocationRelativeTo(null);

Log3.setVisible(true);

}

}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

user.deleteQuestionCustomization();

}

private void jButton16ActionPerformed(java.awt.event.ActionEvent evt) {

user.nextQuestion();

}

private void jButton15ActionPerformed(java.awt.event.ActionEvent evt) {

user.previousQuestion();

}

private void jButton17ActionPerformed(java.awt.event.ActionEvent evt) {

Exception2.dispose();

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

Exception1.dispose();

}

private void jButton18ActionPerformed(java.awt.event.ActionEvent evt) {

Exception3.dispose();

}

private void jButton19ActionPerformed(java.awt.event.ActionEvent evt) {

Exception4.dispose();

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

Stat.dispose();

}

private void jButton14ActionPerformed(java.awt.event.ActionEvent evt) {

CreateQuestions.dispose();

}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

CreateTest.dispose();

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

user.deleteTest();

Log2.setBounds(0, 0, 353, 125);

Log2.setLocationRelativeTo(null);

Log2.setVisible(true);

}

private void jButton20ActionPerformed(java.awt.event.ActionEvent evt) {

JFileChooser fileChooser = new JFileChooser();

int returnValue = fileChooser.showOpenDialog(null);

if (returnValue == JFileChooser.APPROVE\_OPTION) {

File selectedFile = fileChooser.getSelectedFile();

try {

BufferedImage img = ImageIO.read(selectedFile);

image = img;

jPanel17.repaint();

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

private void jMenu7MouseClicked(java.awt.event.MouseEvent evt) {

jframe.setLocationRelativeTo(null);

jframe.setVisible(true);

CreateDate.dispose();

writeLogin.setText("");

jPasswordField1.setText("");

jTextField3.setText("");

jPasswordField2.setText("");

jPasswordField3.setText("");

}

private void jMenu8MouseClicked(java.awt.event.MouseEvent evt) {

Profile.setBounds(0, 0, 650, 290);

Profile.setLocationRelativeTo(null);

Profile.setVisible(true);

CreateDate.dispose();

}

private void jMenu9MouseClicked(java.awt.event.MouseEvent evt) {

TestList.setBounds(0, 0, 600, 310);

TestList.setLocationRelativeTo(null);

TestList.setVisible(true);

CreateDate.dispose();

if (user.getRights()) {

jButton11.setVisible(true);

jButton12.setVisible(true);

jButton13.setVisible(true);

} else {

jButton11.setVisible(false);

jButton12.setVisible(false);

jButton13.setVisible(false);

}

DefaultListModel model = new DefaultListModel();

for (Test test : user.getTestList()) {

if (test.isActive() || user.getRights()) {

model.addElement(test.getName());

}

}

jList1.setModel(model);

if (model.isEmpty()) {

Exception5.setBounds(0, 0, 353, 125);

Exception5.setLocationRelativeTo(null);

Exception5.setVisible(true);

}

}

private void jButton21ActionPerformed(java.awt.event.ActionEvent evt) {

Exception5.dispose();

}

private void jMenu1MouseClicked(java.awt.event.MouseEvent evt) {

jframe.setLocationRelativeTo(null);

jframe.setVisible(true);

TestList.dispose();

writeLogin.setText("");

jPasswordField1.setText("");

jTextField3.setText("");

jPasswordField2.setText("");

jPasswordField3.setText("");

}

private void jButton22ActionPerformed(java.awt.event.ActionEvent evt) {

String firstName = jTextField1.getText();

String lastName = jTextField10.getText();

String eMail = jTextField11.getText();

DB\_User.updateUser(user.getId(), firstName, lastName, eMail);

Log4.setBounds(0, 0, 353, 125);

Log4.setLocationRelativeTo(null);

Log4.setVisible(true);

jTextField1.setText("");

jTextField10.setText("");

jTextField11.setText("");

}

private void jButton23ActionPerformed(java.awt.event.ActionEvent evt) {

jTextField1.setText("");

jTextField10.setText("");

jTextField11.setText("");

}

private void jButton24ActionPerformed(java.awt.event.ActionEvent evt) {

Log1.dispose();

}

private void jButton25ActionPerformed(java.awt.event.ActionEvent evt) {

Log2.dispose();

}

private void jButton26ActionPerformed(java.awt.event.ActionEvent evt) {

Log3.dispose();

}

private void jButton27ActionPerformed(java.awt.event.ActionEvent evt) {

Log4.dispose();

}

private void jButton28ActionPerformed(java.awt.event.ActionEvent evt) {

Profile.setBounds(0, 0, 650, 290);

Profile.setLocationRelativeTo(null);

Profile.setVisible(true);

Log5.dispose();

QuestionPattern.dispose();

}

private void jButton29ActionPerformed(java.awt.event.ActionEvent evt) {

Log5.dispose();

}

private void jButton30ActionPerformed(java.awt.event.ActionEvent evt) {

Log6.dispose();

}

private void jButton31ActionPerformed(java.awt.event.ActionEvent evt) {

Log7.dispose();

}

/\*\*

\* @param args the command line arguments

\*/

private BufferedImage image;

public static PD1\_Rinkevica jframe;

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(PD1\_Rinkevica.class

.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(PD1\_Rinkevica.class

.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(PD1\_Rinkevica.class

.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(PD1\_Rinkevica.class

.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

jframe = new PD1\_Rinkevica();

jframe.setVisible(true);

}

});

}