

**TECHNOLOGIJŲ FAKULTETO**

**INFORMATIKOS IR INŽINERIJOS KATEDRA**

**TRANSPORTO PRIEMONIŲ NUOMOS PROGRAMA**

Kursinis darbas

Informatikos studijų programos

valstybinis kodas 6531BX004

Informatikos studijų krypties

Autorius Jevgenij Čiursin

*(parašas) (data)*

Vadovas doc. Dr. Aleksas Narščius

*(parašas) (data)*

Klaipėda, 2023

**TURINYS**

[iliustracijų sąrašas 3](#_Toc144896258)

[lentelių sąrašas 4](#_Toc144896259)

[įvadas 5](#_Toc144896260)

[1. programų sistemos projektavimas 8](#_Toc144896261)

[1.1. Duomenų bazės projektavimas 8](#_Toc144896262)

[1.2. Naudojami duomenys 10](#_Toc144896263)

[1.2.1. Duomenų objektai 12](#_Toc144896264)

[1.2.2. Duomenų struktūros 15](#_Toc144896265)

[1.3. Programinis projektas 16](#_Toc144896266)

[1.4. Projektavimo šablonai 17](#_Toc144896267)

[2. programų sistemos realizacija 23](#_Toc144896268)

[2.1. JPA realizavimas 23](#_Toc144896269)

[2.2. DB užklausos 24](#_Toc144896270)

[2.3. Algoritmai 25](#_Toc144896271)

[2.4. Grafinė naudotojo sąsaja 25](#_Toc144896272)

[3. Programų sistemos kokybės užtikrinimas 27](#_Toc144896273)

[3.1. Testavimas 27](#_Toc144896274)

[3.2. Kodo versijų kontrolė 28](#_Toc144896275)

[išvados 30](#_Toc144896276)

[literatūra 31](#_Toc144896277)

# iliustracijų sąrašas

**[1 pav.](#_Toc128475047)** [Programos suprojektuota du](#_Toc128475047)

[**[1 pav.](#_Toc128475047)** [Sukurta duomenų bazės klasės modelio diagrama naudojant „MagicDraw“ programinę įrangą 9](#_Toc128475047)](#_Toc144896344)

[**[2 pav.](#_Toc128475047)** [Duomenų bazės „Generic DDL“ modelis sukurtas naudojant „MagicDraw“ programinę įrangą 9](#_Toc128475047)](#_Toc144896345)

[**[3 pav.](#_Toc128475047)** [Transporto priemonių nuomos programos architektūrinis modelis 17](#_Toc128475047)](#_Toc144896346)

[**[4 pav.](#_Toc128475047)** [Duomenų bazės lentelių realizacija serverio aplikacijoje 24](#_Toc128475047)](#_Toc144896347)

[**[5 pav.](#_Toc128475047)** [Registracijos, prisijungimo, slaptažodžio atstatymo langai 26](#_Toc128475047)](#_Toc144896348)

[**[6 pav.](#_Toc128475047)** [„GitHub“ platformos kodo įkėlimų grafikas 29](#_Toc128475047)](#_Toc144896349)

[omenų bazė 8](#_Toc128475047)

# lentelių sąrašas

[**1 lentelė.** Minimalių reikalavimų lentelė 5](#_Toc128475081)

[**2 lentelė.** Vertinimo kriterijų lentelė 7](#_Toc128475082)

# įvadas

Tikslas. Įsisavinti duomenų struktūrų, duomenų bazių, grafinės naudotojo sąsajos programavimą, projektavimo šablonų taikymą, kodo versijų kontrolės, dokumentavimo bei testavimo įrankių naudojimą, programuojant dalykinės srities vientisą taikomąją programą.

Siekiant įgyvendinti numatytą tikslą, keliami šie praktikos uždaviniai:

1. Suprojektuoti programų sistemą:

1.1 Suprojektuoti sistemos pradinius duomenis ir rezultatą;

1.2 Apibrėžti programoje naudojamas duomenų struktūras;

1.3 Aprašyti programinio projekto struktūrą;

1.4 Parinkti ir pritaikyti programavimo šablonus projektuojant architektūrą.

2. Realizuoti programų sistemą:

2.1 Suprogramuoti duomenų įvedimo/išvedimo srautus;

2.2 Suprogramuoti programos skaičiavimų algoritmus;

2.3 Suprogramuoti grafinę naudotojo sąsają;

3. Užtikrinti programų sistemos kūrimo proceso valdymą ir kokybės užtikrinimą:

3.1 Sukurti kodo tikrinimo automatinių testus;

3.2 Naudoti kodo versijų kontrolės įrankius.

Programavimo kursinio darbo kūrimui buvo remiamasi duota minimalių reikalavimų lentele (žr. 1 lentelę).

|  |  |
| --- | --- |
| **Minimalūs reikalavimai:** | **Pildo dėstytojas** |
| Laikomasi kodo **vardinimo taisyklių** |  |
| **Ataskaita** be gramatinių ar formatavimo klaidų |  |
| **Gitlab** pateiktas kodas ir ataskaita |  |
| Yra ataskaita su užpildytais visais duoto šablono skyriais |  |
| Kiekviename ataskaitos skyriuje yra aiškiai nurodyta kurioje kodo dalyje yra realizuotas rezultatas |  |

1. Minimalių reikalavimų lentelė

Taip pat buvo remiamasi ir vertinimo kriterijų lentele (žr. 2 lentelę).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vertinama dalis (skyrius ataskaitoje)** | **Vertė** | **Slenkstinis (5-6)** | **Tipinis (7-8)** | **Puikus (9-10)** |
| **Duomenų bazės projektavimas (1.1)** | 5 % | DB su bent 3 lentelėmis (po min 20 įrašų po min 3 laukus). | DB su bent 4 lentelėmis (po min 20 įrašų po min 3 laukus). Panaudoti kelių tipų ryšiai. | Yra skaitymas iš DB su bent 4 lentelėmis (po min 20 įrašų po min 3 laukus). Panaudoti kelių tipų ryšiai.  Sutvarkytos lietuvių kalbos rašmenų koduotės. |
| **Duomenų objektai (1.2.1)** | 5 % | Dirbama bent su vienu duomenų objektu sudarytu iš min 3 savybių | Dirbama su keliais duomenų objektais arba vienu sudėtiniu duomenų objektu | Dirbama su keliais duomenų objektais ir bent vienas yra sudėtinis duomenų objektas |
| **Duomenų struktūros (1.2.2)** | 10 % | Parinkta viena duomenų struktūra ir pagrįstas jos tinkamumas | Panaudotos kelios duomenų struktūros, apibrėžta sava jų kombinacija. Arba yra galimybė papildyti naujais objektais. | Panaudotos kelios duomenų struktūros, apibrėžta sava jų kombinacija. Ir yra galimybė papildyti naujais objektais. |
| **Programinis projektas (1.3)** | 5 % | Aprašytas programinis projektas ir naudojamos technologijos | Aprašytas programinis projektas ir naudojamos technologijos, nurodytas pilnas projekto architektūrinis modelis | Aprašytas programinis projektas ir naudojamos technologijos, nurodytas pilnas projekto architektūrinis modelis. Aprašyti veikimo algoritmų modeliai. |
| **Projektavimo šablonai (1.4)** | 10 % | Programos kode pritaikyta po 1 kūrimo, struktūrinį bei elgesio projektavimo šabloną | Programos kode pritaikyti 5 kūrimo, struktūrinį bei elgesio projektavimo šablonai | Programos kode pritaikyti 7 (po minimum 2 kūrimo, struktūrinį bei elgesio) projektavimo šablonai |
| **JPA realizavimas (2.1)** | 5 % | Sukonfiguruota JPA, naudojant Hibernate arba alternatyvų karkasą. Sukurta bent viena esybė ir atliekamos CRUD operacijos. | Sukonfiguruota JPA, naudojant Hibernate arba alternatyvų karkasą. Sukurtos bent viena sudėtingesnė esybių struktūra (viena esybė sudaryta iš kelių kitų, viena esybė privalomai turi turėti kitą esybę ir pan.). | Sukonfiguruota JPA, naudojant Hibernate arba alternatyvų karkasą. Sukurtos kelios sudėtingesnės esybių struktūros (viena esybė sudaryta iš kelių kitų, viena esybė privalomai turi turėti kitą esybę ir pan.) |
| **DB užklausos (2.1)** | 7 % | Atliekamos visos CRUD operacijos. | Atliekamos sudėtinės užklausos apimančios kelis parametrus arba kelias DB lenteles | Atliekamos sudėtinės užklausos apimančios kelis parametrus ir kelias DB lenteles |
| **Algoritmai (2.2)** | 8 % | Atliekama viena operacija iš:  Elementų paieška kolekcijoje,  Elementų atranka (filtravimas) kolekcijoje,  Elementų rūšiavimas kolekcijoje | Atliekamos dvi operacijos iš:  Elementų paieška kolekcijoje,  Elementų atranka (filtravimas) kolekcijoje,  Elementų rūšiavimas kolekcijoje | Atliekamos trys operacijos:  Elementų paieška kolekcijoje,  Elementų atranka (filtravimas) kolekcijoje,  Elementų rūšiavimas kolekcijoje |
| **Grafinė naudotojo sąsaja  (2.3)** | 10 % | Yra grafinė naudotojo sąsaja, atvaizduojanti duomenis ir leidžianti manipuliuoti jais. | Yra sudėtinė grafinė naudotojo sąsaja, atvaizduojanti duomenis ir leidžianti manipuliuoti jais. | Yra kelios skirtingos sudėtinės grafinės naudotojo sąsajos (mobile app, web, kt.), atvaizduojančios duomenis ir leidžiančios manipuliuoti jais. |
| **Testavimas (3.1)** | 4 % | Sukurtas kodas testuojamas automatiniais testais (padengimas 20 proc.) | Sukurtas kodas testuojamas automatiniais testais (padengimas 50 proc.). | Sukurtas kodas testuojamas automatiniais testais (padengimas 70 proc.) |
| 3 % | Yra panaudoti bent 3 rūšių assert metodai. | Yra panaudoti bent 4 rūšių assert metodai. | Yra panaudoti bent 5 rūšių assert metodai. |
| 3 % | Yra panaudotos bent 3 rūšių anotacijos. | Yra panaudotos bent 4 rūšių anotacijos. | Yra panaudotos bent 5 rūšių anotacijos. |
| 4 % | Realizuota viena iš testavimo kategorijų:  Išimčių testavimas, Performance testavimas,  Parametrizuoti testai. | Realizuotos dvi iš testavimo kategorijų: Išimčių testavimas, Performance testavimas,  Parametrizuoti testai. | Realizuotos trys iš iš testavimo kategorijų:  Išimčių testavimas, Performance testavimas,  Parametrizuoti testai. |
| **Kodo versijų kontrolė (3.2)** | 8 % | Minimum 25% kassavaitiniai tarpiniai kodo pateikimai | Minimum 50% kassavaitinių tarpinių kodo pateikimų | Minimum 75% kassavaitinių tarpinių kodo pateikimų |
| **Sprendimų pagrindimas (Informacinių šaltinių sąrašas)** | 13 % | Projektuojant ir realizuojant pacituoti bent 5 moksliniai šaltiniai | Projektuojant ir realizuojant pacituoti bent 8 moksliniai šaltiniai | Projektuojant ir realizuojant pacituoti bent 10 mokslinių šaltinių |

1. Vertinimo kriterijų lentelė

# programų sistemos projektavimas

Automobilių nuomos programa – tai aplinka, kur prisiregistravęs asmuo turi galimybę išduoti savo asmeninę transporto priemonę į nuomą bei nuomotis transporto priemonę iš kito žmogaus.

Duomenims saugoti parinkta „MySQL“ reliacinė duomenų bazė. Reliacinės duomenų bazės modelis yra universaliausias iš visų duomenų bazių modelių. Reliacinėje duomenų bazėje duomenys saugomi lentelėse, o duomenys iš jų išrenkami pasirenkant stulpelius ir eilutes, atitinkančius naudotojo nustatytas sąlygas. Lentelės gali būti sujungiamos, kad susidarytų didesnės lentelės su visa reikiama informacija (Gehani, 2011). Tai ne tik palengvina informacijos saugojimą bei gavimą, bet ir leidžia kurti ryšius tarp saugomų subjektų kas yra svarbu šioje sistemoje.

Programai realizuoti buvo pasirinkta aukšto lygio programavimo kalba „Java“, leidžianti kūrėjui rašyti programas, kurios nepriklauso nuo konkretaus kompiuterio tipo. Taip pat aukšto lygio kalbas lengviau skaityti, rašyti ir prižiūrėti (Cosmina, 2021).

Vartotojo aplikacijai gauti duomenis buvo pasirinkta naudoti „Java Spring Boot“ karkasas kuris leidžia užsiimti programos logika nešvaistant laiko ilgam galutinės serverio aplikacijos įdiegimo konfigūracijai (Musib, 2022). Šis karkasas bus naudojamas kurti „REST“ aplikacijų programavimo sąsajai kuri leis atskirti vartotojo aplikaciją nuo serverio duodama galimybę vystyti jas atskirai, bei mažinti serverio apkrovą nesaugant papildomos informacijos apie vartotojų būsenas surištas su sistema (Balaji ir Bartkov, 2021).

Vartotojo aplikacijos grafinei sąsajai realizuoti buvo pasirinkta naudoti „JavaFX“ karkasas kuris leidžia kurti išsamią vartotojo sąsaja, taip pat turi reikiamus įrankius vaizdų, vaizdo įrašų integracijai (Kishori ir Späth, 2022).

## Duomenų bazės projektavimas

Duomenų bazės projektavimui naudojama „MagicDraw“ programinė įranga leidžianti ne tik lengvai projektuoti reliacines duomenų bazės modelius, bet ir generuoti „SQL“ kodą greitam duomenų bazės konfigūravimui (Elvis ir Godbole, 2016). Su šia programinę įrangą kursime duomenų bazės klasių modelį, jį konvertuosime į „Generic DDL“ modelį kurį vėliau naudosime „SQL“ kodo generavimui.

Duomenų bazėje saugosime informaciją susietą su vartotojų paskyromis (prisijungimo duomenys, prie jų prisegtos transporto priemonės, kontaktiniai duomenys ir kt.), taip pat bus saugomos sutartys kurios susies vartotojus su užregistruotomis transporto priemonėmis sistemoje. Neatitikčių bei klaidų šalinimui bus saugomas didelis kiekis transporto priemonių modelių bei jų gamintojų iš kurių vartotojai turės rinktis registruojant transporto priemonę. Nuomos metu pažeidus transporto priemonę įvykis bus fiksuojamas ir dokumentuojamas. Duomenų bazės klasės modelio diagramą galima matyti pav. 1.

Paveikslėlis, kuriame yra diagrama

Automatiškai sugeneruotas aprašymas

1. Sukurta duomenų bazės klasės modelio diagrama naudojant „MagicDraw“ programinę įrangą

Duomenų bazė sudaryta iš 9 lentelių. Panaudoti dviejų tipų ryšiai – Vienas su daugeliu ir daug su daug. Daug su daug ryšiui realizuoti naudojamos „isdavimo\_transporto“ ir „nuomos\_transporto“ tarpinės lentelės kurios susieja vartotojus su užregistruotomis sistemoje transporto priemonėmis.

Taip pat, kad palengvinti „MySQL“ duomenų bazės konfigūraciją klasių modelis yra konvertuojamas į „Generic DDL“ modelį kurį galima matyti pav. 2.

Paveikslėlis, kuriame yra diagrama

Automatiškai sugeneruotas aprašymas

1. Duomenų bazės „Generic DDL“ modelis sukurtas naudojant „MagicDraw“ programinę įrangą

Šiame modelyje yra akivaizdžiau matomas ryšys tarp lentelių bei nurodomi išoriniai raktai kurie vėliau ir bus naudojami susieti saugomus objektus duomenų bazėje.

## Naudojami duomenys

Norint sukurti veikiančią transporto priemonių nuomos programą prireikė pasinaudoti išoriniais šaltiniais transporto priemonių gamintojų bei jų modeliams gauti, kuriuos vėliau naudosime tiek nesutapimų duomenyse šalinimui, tiek paieškoje. Pagrindiniai programoje duomenys yra susiję su vartotojų asmeniniais duomenimis bei su jais susietomis transporto priemonėmis.

Vartotojų asmeniniai duomenys yra registruojami į „MySQL“ duomenų bazę bei vėliau naudojami tiek asmens duomenų atvaizdavime bei prisijungime prie programos.

Vartotojo vardas, pavardė, slapyvardis, slaptažodis, el. paštas bei telefono numeris yra išsaugojami sistemoje po sėkmingos registracijos.

public class UserRegisterRequest implements RequestData {

private String username;

private String email;

private String password;

private String name;

private String surname;

private String telephone;

public UserRegisterRequest(String username, String email, String password, String name, String surname, String telephone) {

this.username = username;

this.email = email;

this.password = password;

this.name = name;

this.surname = surname;

this.telephone = telephone;

}

...

Aukščiau galima matyti, kad yra sudaromas duomenų objektas kuris savyje saugo ankščiau paminėtus vartotojo asmeninius duomenis. Registracijos metu šis objektas yra sukuriamas, apdorojamas ir nusiunčiamas tinklo ryšiu į serverį kur vėliau yra įrašomas į duomenų bazę.

Serveriui gaunant prisiregistravusio vartotojo nurodytus duomenis, jie yra sustruktūrizuojami duomenų objekte.

public class RegisterModel {

@JsonProperty("username")

private String username;

@JsonProperty("email")

private String email;

@JsonProperty("password")

private String password;

@JsonProperty("name")

private String name;

@JsonProperty("surname")

private String surname;

@JsonProperty("telephone")

private String telephone;

..

Duomenims užsikrovus jie yra tvarkingai išsaugojami duomenų bazėje.

Kadangi ši programa yra skirta transporto priemonių nuomai bei išdavimui į nuomą, yra galimybė įregistruoti savo transporto priemonę į sistemą.

public class NewRequest implements RequestData {

private String carColor;

private String carEngine;

private String carEngineC;

private String carFuel;

private String carMaker;

private String carModel;

private String carRun;

private String contractLenght;

private String dayPrice;

public NewRequest(String carColor, String carEngine, String carEngineC, String carFuel, String carMaker, String carModel, String carRun, String contractLenght, String dayPrice) {

this.carColor = carColor;

this.carEngine = carEngine;

this.carEngineC = carEngineC;

this.carFuel = carFuel;

this.carMaker = carMaker;

this.carModel = carModel;

this.carRun = carRun;

this.contractLenght = contractLenght;

this.dayPrice = dayPrice;

}

...

Aukščiau galima matyti duomenų objektą kuris savyje saugo registruojamos transporto priemonės gamintoją, modelį, spalvą, informaciją apie variklį, bei vartotojo nurodytą nuomos kainą ir šios transporto priemonės laikymo sistemoje laiką.

### Duomenų objektai

Objektai yra klasės egzemplioriai. Kiekvienas objektas turi savo būseną, elgseną ir tapatybę. Programuojant, problemas skirstome į objektus, todėl transporto priemonių nuomos programai realizuoti naudojama keletas duomenų objektų funkcionalumo įdiegimui bei palaikymui (Sarcar, 2020).

Programoje svarbiems duomenims saugoti yra naudojamas „Singleton“ programavimo šablonu pagrįstas duomenų objektas.

public class StorageMisc {

private static StorageMisc instance;

private final String sceneStart = "/lt/kvk/i17/chursin\_jevgenij/fxml/scenes/StartScene.fxml";

private final String sceneMain = "/lt/kvk/i17/chursin\_jevgenij/fxml/scenes/MainScene.fxml";

private final String contentLogin = "/lt/kvk/i17/chursin\_jevgenij/fxml/content/LoginContent.fxml";

private final String contentRegister = "/lt/kvk/i17/chursin\_jevgenij/fxml/content/RegisterContent.fxml";

private final String contentForgot = "/lt/kvk/i17/chursin\_jevgenij/fxml/content/ForgotContent.fxml";

private final String contentNew = "/lt/kvk/i17/chursin\_jevgenij/fxml/content/NewContent.fxml";

private StartScene ssState = null;

private AnchorPane startPane = null;

private AnchorPane mainPane = null;

private BasicJsonConverter jsonConverter;

public static StorageMisc getInstance() {

if (StorageMisc.instance == null) {

StorageMisc.instance = new StorageMisc();

}

return StorageMisc.instance;

}

... (setters and getters)

Šiame duomenų objekte yra saugojami tiek duomenys kurie programos veikimo metu yra įrašomi ir vėliau naudojami, tiek lengvesniam pasiekiamumui išsaugotos reikalingų objektų lokacijos.

Grafinės vartotojo sąsajos funkcionalumo realizavimui naudojamas duomenų objektas kuriame tiek saugojami dabartinės scenos duomenys, tiek yra aprašyta būtina logika sąsajos tinkamui darbui.

public class MainController {

private Stage stage;

private static MainController instance;

public static MainController getInstance() {

if (MainController.instance == null) {

MainController.instance = new MainController();

}

return MainController.instance;

}

public void switchStyle(Button button, String styleNew, String styleOld) {

button.getStyleClass().remove(styleOld);

button.getStyleClass().add(styleNew);

}

public void setStage(Stage stage) {

this.stage = stage;

}

public void onStart() {

start();

stage.initStyle(StageStyle.TRANSPARENT);

stage.show();

}

public void onLogin() { openMainScene(); }

public void onExit() {

terminateApp();

}

public void onMinimize() { minimizeApp(); }

public void switchRegister() {

gotoRegister();

}

public void switchLogin() { gotoLogin(); }

public void switchForgot() {

gotoForgot();

}

...

Šis objektas taip pat yra pagrįstas „Singleton“ programavimo šablono principu tačiau iš savęs jis daugiau atlieka programos funkcionalumą negi duomenų saugojimą ir perdavimą.

Taip pat programoje atlikti duomenų siuntimui bei serveryje šių duomenų gavimui yra naudojami duomenų pervedimo objektai (angl. Data Transfer Object) (toliau DPO). DPO leidžia struktūrizuoti gaunamus ar esamus duomenis objektuose, kad juos galima būtų vėliau panaudoti programoje (Noback, 2019). Vartotojui registruojantis prie sistemos, užpildyti duomenys formoje yra struktūrizuojami „UserRegisterRequestFactory“ objekte.

public class UserRegisterRequestFactory implements RequestDataFactory {

private String username;

private String email;

private String password;

private String name;

private String surname;

private String telephone;

public UserRegisterRequestFactory(String username, String email, String password, String name, String surname, String telephone) {

this.username = username;

this.email = email;

this.password = password;

this.name = name;

this.surname = surname;

this.telephone = telephone;

}

@Override

public RequestData createRequest() {

return new UserRegisterRequest(username, email, password, name, surname, telephone);

}

}

Aukščiau pateiktame objekte išsaugoti duomenys yra struktūrizuojami tinkamai tam, kad jie būtų paruošti siuntimui bei šių duomenų lengvam struktūrizavimui serveryje.

Serveryje vartotojo registracijoje nurodyti duomenys yra gaunami tinklo ryšiu ir pasinaudojus „Spring Boot“ programinio karkaso metodais, šie duomenys yra įrašomi į žemiau nurodytą objektą.

public class RegisterModel {

@JsonProperty("username")

private String username;

@JsonProperty("email")

private String email;

@JsonProperty("password")

private String password;

@JsonProperty("name")

private String name;

@JsonProperty("surname")

private String surname;

@JsonProperty("telephone")

private String telephone;

Serveryje naudojantis šiuo objektų, duomenys yra apdorojami, jiems esant teisingiems jie yra išsaugojami duomenų bazėje.

### Duomenų struktūros

Duomenų struktūra - tai efektyvus duomenų elementų saugojimo ir organizavimo kompiuterio atmintyje būdas. Duomenys - tai reikšmė arba reikšmių rinkinys. Struktūra reiškia duomenų organizavimo būdą. Matematinis arba loginis duomenų atvaizdavimas atmintyje vadinamas duomenų struktūra. Duomenų struktūros tikslas - efektyviai saugoti, gauti ir atnaujinti duomenis (Malhotra N. ir Malhotra D., 2020). Programoje naudojamos kelios duomenų struktūros tiek paprastam duomenų saugojimui, papildymui, tiek duomenų saugojimui ir tiksliniui duomenų gavimui.

Duomenų persiuntimui tinklo ryšiu yra kuriami „HTTP“ užklausų objektai kurie turi savyje daug parametrų. Šie objektai yra iš anksto sukuriami jų vėlesniam panaudojimui. Šių objektų saugojimui bei tiksliam gavimui yra naudojama „Map“ duomenų struktūra kurioje saugant objektą nurodome identifikacinį raktą šiam objektui gauti.

public class RequestInvoker {

private Map<String, Command> requests;

public RequestInvoker () {

requests = new HashMap<>();

}

public void addRequest(String name, Command request) {

if(requests.get(name) == null) {

requests.put(name, request);

} else {

System.out.println("Such request name already exists!");

}

}

public CloseableHttpResponse executeRequest(String name, String data) {

CloseableHttpResponse response = null;

try {

response = requests.get(name).execute(data);

} catch (Exception e) {

System.out.println("Request with such name doesn't exist!");

System.out.println(e);

}

if (response != null) {

return response;

}

return null;

}

}

Sukuriant ir saugant „HTTP“ užklausos objektą nurodome identifikacinį raktą „name“ kaip tai yra nurodyta „addRequest“ metode. Norint panaudoti objektą nurodome identifikacinį raktą ir jam egzistuojant su juo yra dirbama „executeRequest“ metode.

Serveryje duomenims saugoti į duomenų bazę yra naudojama „List“ duomenų struktūra, kadangi esant dideliam užklausų kiekiui, gali prireikti saugoti kelis objektus vienu metu.

public Vartotojas saveToDatabase(RegisterModel data){

List<Vartotojas> domainObjects = mapData(data);

vartotojas.saveAll(domainObjects);

}

Aukščiau galima matyti, kad „Vartotojas“ objektai yra saugojami duomenų struktūroje ir vėliau ši duomenų struktūra yra panaudojama sąveikoje su duomenų baze.

## Programinis projektas

Transporto priemonių nuomos programos realizacijai yra naudojamos patikimos technologijos bei įrankiai kurie užtikrina programos veikimą.

Programoje naudojami duomenys susieti su vartotojais, jų transporto priemonėmis yra saugojami „MySQL“ reliacinėje duomenų bazėje.

Duomenų perdavimui, tarp vartotojo programos ir serverio, tinklo ryšiu yra naudojamas tiek „Spring boot“ programavimo karkasas „REST API“ realizacijai kuri priima „HTTP“ užklausas serveryje, tiek „ApacheHTTP“ biblioteka šių užklausų kūrimui ir siuntimui iš vartotojo programos pusės.

Kuriant šią programą buvo naudojama „IntelliJ“ aparatinė sąsaja kurios pagalba galima buvo pereiti iš vienos programavimo veiklos dalies į kitą, nereikėjo rūpintis „JVM“ redagavimo, kompiliavimo ir paleidimo mechanikomis (Burd, 2021).

Programos veikimo komponentų architektūrinį modelį galima matyti 3 pav.

Paveikslėlis, kuriame yra tekstas, diagrama, ekrano kopija, Šriftas

Automatiškai sugeneruotas aprašymas

1. Transporto priemonių nuomos programos architektūrinis modelis

Atsižvelgus į programos architektūrinį modelį galima matyti, kad grafinės vartotojo sąsajos, vartotojo aplikacijos logikos bei “Apache HTTP” užklausų realizacijos komponentai priklauso vartotojo aplikacijai, o “Spring boot REST API” realizacija, serverio aplikacijos logika ir “MySQL” duomenų bazės komponentai priklauso serverio aplikacijai.

## Projektavimo šablonai

Programinės įrangos inžinerijoje projektavimo šablonai – tai šablonai, kurie atsiranda iš gerai suprojektuotų program, skirtų tipinėms problemoms, kurios įprastai kartojasi, spręsti. Projektavimo šablonus taip pat galima laikyti geriausia praktika, taikoma siekiant veiksmingai įveikti programinės įrangos projektavimo problemas (Mehra, 2021).

Transporto priemonių nuomos programoje buvo panaudoti 5 projektavimo šablonai kurie yra kūrimo ir elgesio tipo.

Reikalingų duomenų bei objektų saugojimui bei gavimui programos veikimo metu buvo panaudotas „Singleton“ kūrimo projektavimo šablonas. Dėka to, kad šis projektavimo šablonas apibrėžia klasę kaip objektą kuris yra sukuriamas tik vieną kartą per visą programos veikimo laikotarpį, jame galima saugoti reikšmes, kurių gali vėliau prireikti, nesibaiminant jų pametimo (Sarcar, 2019).

public class StorageMisc {

private static StorageMisc instance;

private final String sceneStart = "/lt/kvk/i17/chursin\_jevgenij/fxml/scenes/StartScene.fxml";

private final String sceneMain = "/lt/kvk/i17/chursin\_jevgenij/fxml/scenes/MainScene.fxml";

private final String contentLogin = "/lt/kvk/i17/chursin\_jevgenij/fxml/content/LoginContent.fxml";

private final String contentRegister = "/lt/kvk/i17/chursin\_jevgenij/fxml/content/RegisterContent.fxml";

private final String contentForgot = "/lt/kvk/i17/chursin\_jevgenij/fxml/content/ForgotContent.fxml";

private final String contentNew = "/lt/kvk/i17/chursin\_jevgenij/fxml/content/NewContent.fxml";

private StartScene ssState = null;

private AnchorPane startPane = null;

private AnchorPane mainPane = null;

private BasicJsonConverter jsonConverter;

public static StorageMisc getInstance() {

if (StorageMisc.instance == null) {

StorageMisc.instance = new StorageMisc();

}

return StorageMisc.instance;

}

Toliau duomenų saugojimo bei gavimo logika (setters/getters)

...

Naudojantis šiuo projektavimo šablonu saugojama ne tik svarbius objektus, bet ir paprastesnes reikšmes kurias galima pakeisti esant reikiamybei.

„HTTP“ užklausų objektų kūrimui ir konfigūravimui buvo panaudotas „Builder“ kūrimo projektavimo šablonas. Šis projektavimo šablonas yra naudingas kai objektas yra sukuriamas iš skirtingų dalių (Sarcar, 2019).

public class HttpRequestBuilder {

private String userID;

private String requestBody;

private String url;

private HttpUriRequest request = null;

public HttpRequestBuilder setUrl(String url) {

this.url = url;

return this;

}

public HttpRequestBuilder setUserID(String userID) {

this.userID = userID;

return this;

}

public HttpRequestBuilder requestBody(String requestBody) {

this.requestBody = requestBody;

return this;

}

public void setGet() {

request = new HttpGet(url);

}

public void setPost() {

request = new HttpPost(url);

}

public void setDelete() {

request = new HttpDelete(url);

}

public void setPut() {

request = new HttpPut(url);

}

public HttpUriRequest build() {

request.setHeader("Content-Type", "application/json");

if (requestBody != null && request instanceof HttpEntityEnclosingRequestBase) {

((HttpEntityEnclosingRequestBase) request).setEntity(new StringEntity(requestBody, "UTF-8"));

}

return request;

}

}

Čia užklausos konfigūraciniai parametrai yra nustatomi per metodus ir galutinis sukonfigūruotas objektas yra grąžinamas jo vėlesniam naudojimui.

Tinkamam duomenų struktūrizavimui prieš išsiuntimą į serverį naudojamas „Factory“ projektavimo šablonas kurti objektams.

public class UserLoginRequestFactory implements RequestDataFactory {

private String username;

private String password;

public UserLoginRequestFactory(String username, String password){

this.username = username;

this.password = password;

}

@Override

public RequestData createRequest() {

return new UserLoginRequest(username, password);

}

}

Nustačius reikšmes yra sukuriamas objektas su duomenų nustatymo ir grąžinimo metodais vėlesniam modifikavimui bei su duomenų struktūra kuri yra tinkama išsiuntimui į serverį.

Grafinės vartotojo sąsajos dizaino keitimuisi vartotojui sąveikaujant yra naudojamas „State“ projektavimo šablonas kuris keičia tai, kas yra atvaizduojama ekrane, bei programos veikimo principus.

public class StartScene implements StartSceneState {

<...>

public StartScene(Button register, Button login, FontAwesomeIconView regIcon, FontAwesomeIconView logIcon) {

registerState = new RegisterState(this);

loginState = new LoginState(this);

forgotState = new ForgotState(this);

currentState = registerState;

registerBtn = register;

loginBtn = login;

lastBtn = register;

this.regIcon = regIcon;

this.logIcon = logIcon;

}

@Override

public void changeState(Button clickedBtn) {

if (clickedBtn == null) {

lastState = getIntValue(currentState);

currentState = forgotState;

currentState.changeState(null);

}

else if(lastBtn == clickedBtn && currentState != forgotState) {

}

else if(lastBtn == clickedBtn && currentState == forgotState) {

lastState = getIntValue(currentState);

currentState = getState(clickedBtn);

currentState.changeState(clickedBtn);

}

else {

lastState = getIntValue(currentState);

lastBtn = clickedBtn;

currentState = getState(clickedBtn);

currentState.changeState(clickedBtn);

}

}

...

Aukščiau galima matyti, kad vartotojo grafinė sąsaja turi 3 būsenas. Priklausant nuo mygtukų paspaudimų, būsenos keičiasi į registravimosi, prisijungimo bei užmiršto slaptažodžio.

Lengvam sukurtų „HTTP“ užklausų pasiekimui bei jų suaktyvinimui yra naudojamas „Command“ projektavimo šablonas kuris leidžia nurodyti paleidimo logiką atskirai kiekvienam saugojamui objektui.

public class RequestInvoker {

private Map<String, Command> requests;

public RequestInvoker () {

requests = new HashMap<>();

}

public void addRequest(String name, Command request) {

if(requests.get(name) == null) {

requests.put(name, request);

} else {

System.out.println("Such request name already exists!");

}

}

public CloseableHttpResponse executeRequest(String name, String data) {

CloseableHttpResponse response = null;

try {

response = requests.get(name).execute(data);

} catch (Exception e) {

System.out.println("Request with such name doesn't exist!");

System.out.println(e);

}

if (response != null) {

return response;

}

return null;

}

Aukščiau galima matyti, kad skirtingi objektai yra išsaugojami „Map“ duomenų struktūroje iš kurios objekto ištraukimas vykdomas nurodant identifikacinį raktą. Egzistuojant objektui jo paleidimo logika yra suaktyvinama.

# programų sistemos realizacija

Šio skyriaus tikslas yra aprašyti transporto priemonių nuomos programos realizaciją kuri apima duomenų bazės struktūros projektavimą, naudotus skaičiavimo algoritmus ir sukurtą grafinę naudotojo sąsają.

## JPA realizavimas

JPA – tai metodų rinkinys kurio pagalba realizuojama programa ar sistema yra apjungiama su duomenų baze duomenims saugoti bei gauti (Malhotra, 2019). Transporto priemonių nuomos programoje buvo įvykdyta JPA realizacija serverio komunikacijai su duomenų baze.

Kiekviena duomenų bazėje esanti lentelė yra aprašyta serverio aplikacijoje kur yra nurodomi saugojami duomenys bei jų struktūra.

@Entity

@ToString

public class Vartotojas {

@Id

@Column(nullable = false, updatable = false)

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

@Column

private String name;

@Column

private String surname;

@Column

private String login;

@Column

private String password;

public Integer getId() {

return id;

}

public void setId(final Integer id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(final String name) {

this.name = name;

}

public String getSurname() {

return surname;

}

public void setSurname(final String surname) {

this.surname = surname;

}

public String getLogin() {

return login;

}

public void setLogin(final String login) {

this.login = login;

}

public String getPassword() {

return password;

}

public void setPassword(final String password) {

this.password = password;

}

}

Aukščiau galima matyti „Vartotojas“ objektą kuris apibrėžia duomenų bazėje esančią „vartotojas“ lentelę. Čia nurodyti vartotojo vardas, pavardė, slapyvardis, slaptažodis, šių duomenų įrašymo bei nuskaitymo metodai.

Visų lentelių objektų realizaciją galima matyti 4 pav.

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Šriftas

Automatiškai sugeneruotas aprašymas

1. Duomenų bazės lentelių realizacija serverio aplikacijoje

## DB užklausos

Duomenų bazės užklausoms realizuoti yra naudojami JPA metodai duomenų saugojimui, keitimui bei naikinimui.

Kiekvienai lentelei yra kuriama sąsaja kurioje gali būti pridėta papildoma verslo logika bei yra galimybė panaudoti bazinius metodus susietus su duomenų lentelės interaktyvumu.

@Repository

public interface VartotojasRepository extends JpaRepository<Vartotojas, Integer> {

Vartotojas findByLogin(String login);

}

Aukščiau galima matyti sąsaja kuri yra apjungta su JPA teikiama klase. Taip pat yra nurodyta papildoma verslo logika surišta su vartotojo paieška duomenų bazėje pagal vartotojo suteikiamą slapyvardį.

Prisiregistravusio vartotojo duomenims išsaugoti naudojama JPA karkaso pavidalu sukurta klasė kuri apibrėžia duomenų bazėje esančia lentelę ir šio karkaso teikiamu „saveAll“ metodu. Duomenis yra sukeliami į klasę ir ši klasė yra išsaugojama.

public Vartotojas saveToDatabase(RegisterModel data){

List<Vartotojas> domainObjects = mapData(data);

vartotojas.saveAll(domainObjects);

}

Aukščiau galima matyti, kad duomenys („data“) yra priskiriami prie „Vartotojas“ klasės ir vėliau visi „Vartotojas“ klasės objektai yra išsaugojami su „saveAll“ metodu.

## Algoritmai

Transporto priemonių nuomos programoje nebuvo realizuoti jokie paieškos algoritmai.

## Grafinė naudotojo sąsaja

Transporto priemonių nuomos programos grafinės naudotojo sąsajos realizacijai buvo naudota “JavaFX” biblioteka. Grafinė naudotojo sąsaja susideda iš duomenų įvedimo laukų, mygtukų, aprašų ir kt.

Prisijungimo, registracijos bei užmiršto slaptažodžio langai yra atvaizduojami vienoje scenoje „State“ projektinio šablono pagalba (5 pav.).

Paveikslėlis, kuriame yra tekstas, elektronika, ekrano kopija, programinė įranga

Automatiškai sugeneruotas aprašymasPaveikslėlis, kuriame yra tekstas, elektronika, ekrano kopija, programinė įranga

Automatiškai sugeneruotas aprašymasPaveikslėlis, kuriame yra tekstas, ekrano kopija, Šriftas, Tinklalapis

Automatiškai sugeneruotas aprašymas

1. Registracijos, prisijungimo, slaptažodžio atstatymo langai

Čia suvedus atitinkamus duomenis, galima sąveikauti su programos duomenų bazę tolimesniam jos naudojimui.

# Programų sistemos kokybės užtikrinimas

Programinės sistemos kokybės užtikrinimas yra svarbus procesas kuris apima daugybę veiksmų įskaitant kodo testavimą, analizę bei kitus svarbius procesus. Sistemos kokybės užtikrinimas yra atliekamas tam, kad sumažinti programoje esantį klaidų kiekį bei palaikyti aukštą kodo kokybę. Dėka to yra užtikrinamas programos patikimumas, stabilumas bei kliento reikalavimų atitikimas.

## Testavimas

Programos kodo kokybei užtikrinti bei klaidų likvidavimui buvo naudoti “JUnit” testavimo įrankiai kurie leidžia kurti testavimo vienetus klasių metodų testavimams (Tudose, 2020).

Duomenų bazės duomenų įrašymo, atnaujinimo bei naikinimo metodų testavimui naudojamos papildomas anotacijos nurodančios, kad tai yra modifikuotas “Spring boot” karkaso testavimas.

@SpringBootTest

class VartotojasRepositoryTest {

@Autowired

private VartotojasRepository vartotojasRepository;

@Test

void saveMethod() {

//create Entity

Vartotojas vartotojas = new Vartotojas();

vartotojas.setName("Jevgenij");

vartotojas.setSurname("Chursin");

vartotojas.setLogin("augis151");

vartotojas.setPassword("test123");

//save Entity

Vartotojas savedEntity = vartotojasRepository.save(vartotojas);

//display Entity info

System.out.println(savedEntity.toString());

}

Su “@SpringBootTest” anotacija nurodome, kad šie testai turi būti paleidžiami naudojantis karkaso įrankiais.

Su „@Autowired“ anotacija mes automatiškai priskiriame duomenų bazės lentelės objektą jo tolimesniam naudojimui testuose.

Su „@Test“ anotacija mes nurodome, kad žemiau esantis metodas tai yra bazinis testas.

Vartotojo programoje, kad patikrinti ar “HTTP” užklausų objektai yra tinkamai sukuriami bei ar veikia „Command“ projektavimo šablonas yra naudojamos kelios rūšys „assert“ metodų.

@Test

public void testAddRequest() {

RequestInvoker invoker = new RequestInvoker();

Command request = new GetRequest("https://test.url", null);

invoker.addRequest("TestRequest", request);

assertNotNull(invoker.getRequest("TestRequest"));

}

@Test

public void testAddRequestDuplicateName() {

RequestInvoker invoker = new RequestInvoker();

Command request1 = new GetRequest("https://test.url", null);

Command request2 = new GetRequest("https://test.url", null);

invoker.addRequest("TestRequest", request1);

invoker.addRequest("TestRequest", request2);

// Assert that the second addition fails

assertNull(invoker.getRequest("TestRequest"));

}

@Test

public void testEqualsRequestSetData() {

RequestInvoker invoker = new RequestInvoker();

Command request = new GetRequest("https://test.url", "test");

invoker.addRequest("TestRequest", request);

assertEquals("test", invoker.getRequest("TestRequest").getID());

}

„assertNotNull“ – patikrina ar gautas rezultatas egzistuoja, jei jis egzistuoja, testas yra įvykdomas sėkmingai.

„assertNull“ – patikrina ar gautas rezultatas neegzistuoja, jei jis egzistuoja, testas nėra sėkmingai įvykdomas.

„assertEquals“ – patikrina ar duotos dvi reikšmės yra tokios pačios, jei jos atitinka vienas kitą, testas yra įvykdomas sėkmingai.

## Kodo versijų kontrolė

Kodo versijų kontrolei atlikti buvo pasirinkta “GitHub” platforma, jos pagalba programos rašytas kodas buvo saugomas versijomis kurias galima būtų vėliau atkurti to prireikus bei tai davė galimybę užkrauti projektą kitose mašinose esant reikiamumui.

Programos kūrimo metu 25 procentai kassavaitinių kodo įkėlimų nebuvo pasiekti (6 pav.).

Paveikslėlis, kuriame yra ekrano kopija, linija

Automatiškai sugeneruotas aprašymas

1. „GitHub“ platformos kodo įkėlimų grafikas

Šio projekto adresas “GitHub” platformoje yra - <https://github.com/Put-in-228/kursnarscius>

# išvados

1. Atlikus programų sistemos projektavimą buvo pagilintos žinios projektavimo įrankių naudojime, patobulėjo informacijos paieškos įgūdžiai renkantis technologijas projekto funkcionalumo realizacijai. Integruojant programavimo šablonus į projektą išplatėjo supratimas apie šių šablonų svarbą bei jų taikymo principus.
2. Atlikus programų sistemos įgyvendinimą, buvo tiek susipažinta su „Spring boot“ karkaso, „JavaFX“ teikiamais įrankiais. Buvo susipažinta su „JPA“ teikiama duomenų bazės integracija į aplikaciją naudojant „Java“ klasių integravimus. Patobulėjo programavimo su „Java“, bei grafinės vartotojo sąsajos projektavimo ir realizavimo įgūdžiai.
3. Atlikinėjant programos testavimą, bei versijavimą buvo susipažinta su teikiamais modifikuotais testavimo įrankiais iš „Spring Boot“ programavimo karkaso bei patobulėjo įgūdžiai „JUnit“ vienetinių testų rašyme bei pritaikyme. Naudojant „GitHub“ platformą projekto versijavimui buvo toliau lavinami įgūdžiai projektų kūrime bei taisyklingame kodo talpinime.

# literatūra

1. Gehani N. (2011). The Database Book: Principles & Practice Using MySQL, Second Edition. Silicon Press
2. Cosmina I. (2021). Java 17 for Absolute Beginners: Learn the Fundamentals of Java Programming. Apress.
3. Musib S. (2022). Spring Boot in Practice. Manning Publications.
4. Balaji V. ir Bartkov M. (2021). Spring REST: Building Java Microservices and Cloud Applications. Apress.
5. Kishori S. ir Späth P. (2022). Learn JavaFX 17: Building User Experience and Interfaces with Java, 2nd Edition. Apress.
6. Elvis C. F. ir Godbole S. (2016). Database Systems: A Pragmatic Approach, Second Edition. Apress.
7. Sarcar V. (2020). Interactive Object-Oriented Programming in Java: Learn and Test Your Programming Skills, Second Edition. Apress,
8. Noback M. (2019). Object Design Style Guide. Manning Publications.
9. Malhotra D., ir Malhotra N. (2020). Data Structures and Program Design Using Java: A Self-Teaching Introduction. Mercury Learning.
10. Burd B. (2021). Beginning Programming with Java For Dummies, 6th Edition. John Wiley & Sons.
11. Mehra L. (2021). Software Design Patterns for Java Developers: Expert-Led Approaches to Build Re-Usable Software and Enterprise Applications. BPB Publications.
12. Sarcar V. (2019). Java Design Patterns: A Hands-On Experience with Real-World Examples, Second Edition. Apress.
13. Malhotra D. (2019). Rapid Java Persistence and Microservices: Persistence Made Easy Using Java EE8, JPA and Spring. Apress.
14. Tudose C. (2020). JUnit in Action, Third Edition. Manning Publications.