



VILNIAUS UNIVERSITETAS  
MATEMATIKOS IR INFORMATIKOS FAKULTETAS  
STUDIJŲ PROGRAMA: INFORMACINĖS TECHNOLOGIJOS

Baigiamasis bakalauro darbas

**CRUD funkcijų generatorius Laravel karkasui**  
CRUD functions generator for Laravel framework

Atliko:

Rafal Kadzevič

VU el. p.: rafal.kadzevic@mif.stud.vu.lt

Vadovas:

dr. Joana Katina

Vilnius  
2023

# Turinys

|                                                                                 |           |
|---------------------------------------------------------------------------------|-----------|
| <b>Sutartinis terminų žodynas</b>                                               | <b>4</b>  |
| <b>Santrauka</b>                                                                | <b>5</b>  |
| <b>Summary</b>                                                                  | <b>6</b>  |
| <b>Ivydas</b>                                                                   | <b>7</b>  |
| <b>1. Sukurtų sprendimų analizė</b>                                             | <b>9</b>  |
| 1.1. Plačiai naudojamų Laravel karkaso CRUD generavimo paketų analizė . . . . . | 9         |
| 1.1.1. <i>infyomlabs/laravel-generator</i> paketas . . . . .                    | 9         |
| 1.1.2. <i>roland/crud</i> paketas . . . . .                                     | 10        |
| 1.1.3. <i>crestapps/laravel-code-generator</i> paketas . . . . .                | 11        |
| 1.1.4. <i>larafast/fastapi</i> paketas . . . . .                                | 12        |
| 1.2. Sukurtų sprendimų analizės apibendrinimas . . . . .                        | 13        |
| <b>2. Sistemos modelis</b>                                                      | <b>14</b> |
| 2.1. Sistemos funkcionalumas . . . . .                                          | 14        |
| 2.1.1. Funkciniai reikalavimai . . . . .                                        | 14        |
| 2.1.2. Nefunkciniai reikalavimai . . . . .                                      | 15        |
| 2.2. UML diagramos . . . . .                                                    | 15        |
| 2.2.1. Naudojimo atvejų diagrama . . . . .                                      | 15        |
| 2.2.2. Klasių diagrama . . . . .                                                | 17        |
| 2.2.3. Sekos diagrama . . . . .                                                 | 18        |
| 2.2.4. Veikos diagrama . . . . .                                                | 19        |
| 2.2.5. Sistemos navigacijos diagrama . . . . .                                  | 21        |
| 2.2.6. Duomenų bazės schema . . . . .                                           | 21        |
| <b>3. Sistemos programinė realizacija</b>                                       | <b>23</b> |
| 3.1. Naudojami įrankiai ir technologijos . . . . .                              | 23        |
| 3.2. Struktūros aprašymas . . . . .                                             | 23        |
| 3.3. CRUD generavimo aprašymas . . . . .                                        | 24        |
| 3.4. Migracijos generavimo aprašymas . . . . .                                  | 24        |
| 3.5. Modelio generavimo aprašymas . . . . .                                     | 25        |
| 3.6. Valdiklio generavimo aprašymas . . . . .                                   | 25        |
| 3.7. Vaizdų generavimo aprašymas . . . . .                                      | 26        |
| 3.8. Modelių sąryšių valdiklio aprašymas . . . . .                              | 26        |
| 3.9. Konfigūruojamo puslapiavimo aprašymas . . . . .                            | 27        |
| 3.10. Paketo publikavimas . . . . .                                             | 27        |
| <b>Išvados</b>                                                                  | <b>28</b> |
| <b>Ateities darbų planas</b>                                                    | <b>29</b> |
| <b>Literatūros šaltiniai</b>                                                    | <b>30</b> |

|                                                             |           |
|-------------------------------------------------------------|-----------|
| <b>Priedai</b>                                              | <b>31</b> |
| <b>A. Paketo paruoštiniai failai</b>                        | <b>31</b> |
| A.1. Valdiklio paruoštinis failas . . . . .                 | 31        |
| A.2. Migracijos paruoštinis failas . . . . .                | 33        |
| A.3. Sąryšių migracijos paruoštinis failas . . . . .        | 34        |
| A.4. Modelio paruoštinis failas . . . . .                   | 34        |
| A.5. Pagrindinio vaizdo failo paruoštinis failas . . . . .  | 36        |
| A.6. Formos paruoštinis failas . . . . .                    | 37        |
| A.7. Informacijos vaizdo failo paruoštinis failas . . . . . | 37        |
| <b>B. Sistemos valdikliai</b>                               | <b>38</b> |
| B.1. CrudController valdiklis . . . . .                     | 38        |
| B.2. ModelRelationshipController valdiklis . . . . .        | 42        |
| <b>C. Paketo programinio kodo pavyzdžiai</b>                | <b>48</b> |
| C.1. Duomenų perdavimo objektas (DTO) . . . . .             | 48        |
| C.2. CrudConfig.php failas . . . . .                        | 48        |

## Sutartinis terminų žodynas

**CRUD (angl. Create, Read, Update, Delete)** – akronimas keturių pagrindinių funkcijų dirbant su duomenimis: kurti, skaityti, atnaujinti, ištrinti.

**PHP** – interpretuojama, atviro kodo programavimo kalba, skirta kurti tinklalapius ir žiniatinklio programas realiuoju laiku.

**HTML** – žymėjimo kalba, naudojama kuriant hipertekstinius dokumentus.

**CSS** – yra stiliaus aprašymo kalba, naudojama HTML dokumento vaizdo dizaino ir stiliaus apibūdinimui.

**UML** – formali kalba, naudojama įvairių tipų sistemų modeliavimui.

**SOLID** – akronimas, reiškiantis penkis pagrindinius projektavimo principus: vienos atsakomybės principą, atviro uždarymo principą, Liskovo pakeitimo principą, sąsajos atskyrimo principą ir priklausomybės inversijos principą.

**PSR (angl. PHP Standards Recommendations)** – akronimas, reiškiantis pagrindinės PHP programavimo kalbos kodo rašymo stiliaus standartus ir rekomendacijas.

**MVC (angl. Model, View, Controller)** – akronimas, reiškiantis programinės įrangos kūrimo modelį, kuriame programinis kodas skirstomas į modelius, valdiklius ir vaizdo failus.

**ORM (angl. Object-Relational Mapping)** – programinė įranga, leidžianti adresuoti duomenų objektus, juos pasiekti ir manipuluoti nenaudojant reliacinių duomenų bazių užklausų.

**DTO (angl. Data Transfer Object)** – akronimas, objektiniame programavime reiškiantis klasę, kuri naudojama tik duomenų perdavimui tarp metodų ir klasių.

**JSON (angl. JavaScript Object Notation)** – failo formatas, skirtas duomenims saugoti ir transportuoti, dažnai naudojamas, kai duomenys siunčiami iš serverio į tinklalapį.

**API (angl. Application Programming Interface)** - sąsaja, leidžianti dviem programinės įrangos komponentams komunikuoti, naudojant apibrėžimų ir protokolų rinkinius.

## Santrauka

Šiame darbe aprašomas CRUD funkcijų kūrimo paketas Laravel karkasui, detali generatoriaus analizė ir realizavimo procesas. Darbo tikslas – sukurti CRUD funkcijų generavimo paketą Laravel karkasui, kuris būtų viešai prieinamas. Pagrindiniai darbo uždaviniai yra išanalizuoti esamus CRUD generavimo paketus, atrasti netikslumus, trūkumus ir privalumus. Analizės metu buvo suformuluoti ir surinkti pagrindiniai sistemos uždaviniai ir aprašytas teorinis sistemos modelis. Pasiektas darbo rezultatas yra sukurtas ir visiems viešai prieinamas Laravel karkaso CRUD generatoriaus paketas, gebantis generuoti grafinę sąsają, kurioje naudotojai gali kurti modelius, valdiklius, migracijas ir vaizdo failus. Be to, paketas palaiko kintamųjų validaciją ir turi keturias pagrindines funkcijas: kurti, skaityti, atnaujinti, ištrinti. Toks paketas leidžia naudotojams generuoti dažnai perteklines operacijas ir funkcijas tik su grafine sąsaja, be įsiterpimo į programinį kodą.

# Summary

## **CRUD functions generator for Laravel framework**

This paper describes a universal package for creating CRUD functions for the Laravel framework, a detailed analysis of the generator, and the implementation process. The aim of the work is to analyze the existing CRUD generator packages, to discover inaccuracies, advantages and disadvantages, and based on the conclusions of the analysis to create an optimal and universal CRUD functions generator package, which will be publicly available to everyone by using package manager called Composer. During the analysis, the main tasks of the system were formulated and collected, and the theoretical model of the system was described. The achieved result of the work is a CRUD generator package for the Laravel framework developed and publicly available to all, capable of generating a graphical interface where users can create a model, manage and migrate. In addition, the package supports variable validation, and has four basic functions: create, read, update, continue. Such a package will allow users to generate often redundant operations and functions only with a graphical interface, without intervening in the program code.

## Ivyadas

Pasaulis nuolat keičiasi, kartu su juo keičiamės ir mes, žmonės, ir visa sistema, kurioje esame įpratę gyventi. Šiais laikais pasaulis juda link kompiuterizacijos, o tai daro daug pokyčių įvairiose srityse. Verslas ir rinka yra geras pavyzdys subjekto, kuris per pastaruosius 30 metų patyrė daug pokyčių dėl kompiuterizacijos. Senasis verslo modelis negali patenkinti realių žmonių poreikių ir reikalavimų, todėl reikia diegti naujus sprendimus. Vienas iš tų naujų sprendimų yra duomenų bazės naudojimas verslo modelyje. Iš pirmo žvilgsnio šis sprendimas turi daug privalumų ir gali atrodyti idealiai lyginant su senu modeliu, kuriame visi duomenys buvo saugomi popieriuje. Pagrindiniai duomenų bazės naudojimo pranašumai yra tai, kad vienoje vietoje gali būti saugomi dideli duomenų kiekiai, duomenų bazėse galima ieškoti ir rūšiuoti, todėl reikiamus duomenis galima rasti greitai ir lengvai, o duomenų struktūra yra plečiama ir gali būti keičiama keičiantis reikalavimams. Papildomai duomenų bazės gali užtikrinti, kad įrašyti duomenys yra teisingi arba vientisi. Duomenų bazės taip pat palaiko aukštą saugumo lygį, leidžia valdyti prieigą, todėl naudotojai gali turėti skirtingas teises, pavyzdžiui, kai kurie naudotojai gali tik skaityti duomenis. Duomenys taip pat gali būti suskirstyti į segmentus, kad naudotojai galėtų pasiekti tik tam tikras dalis, pavyzdžiui, naudotojas gali skaityti darbuotojo vardą ir pavardę bei adresą, bet ne informaciją apie atlyginimą. Visa tai, kas aprašyta aukščiau, leidžia manyti, kad duomenų bazės yra puikus sprendimas, kurį suteikia kompiuterizacija, tačiau iš kitos pusės yra vienas svarbus aspektas – duomenų bazės priežiūrai ir duomenų apdorojimui reikalingi aukšto išsilavinimo specialistai. Kyla klausimas, kaip supaprastinti duomenų apdorojimo procesą ir leisti paprastiems darbuotojams dirbti su duomenų bazės duomenimis be galimybės padaryti rimtų klaidų. Sprendimas, kurį dažnai turi įmonės, yra sukurti paprastą grafinę duomenų apdorojimo sąsają. Tokia sąsaja turi leisti naudotojams lengvai kurti naujus įrašus duomenų bazėje, matyti įrašų sąrašą, atnaujinti senus ir ištrinti nereikalingus. Papildomai vaizdinė sąsaja turi patikrinti įvestus duomenis, patvirtinti, kad nėra klaidų ir informuoti naudotoją apie tai. Vaizdinių sąsajų kūrimas tapo įprasta darbo rutina daugumai programuotojų, tai yra lengvas, bet dažnai perteklinis ir daug laiko užimantis darbas. Dėl minėtos priežasties buvo nuspręsta sukurti universalų įrankį, Laravel karkaso funkcijų paketą, kuris leistų greitai ir saugiai generuoti sąsajas, skirtas paprastiems darbuotojams. Toks paketas leidžia naudotojui pasitelkiant grafinę sąsają automatiškai sukurti modelius, migracijas ir valdiklius. Šis sprendimas suteikia galimybę paprastiems naudotojams prisijungti prie prietaisų skydelio ir sukurti visas reikalingas CRUD dalis be jokių programavimo įgūdžių. Papildomai CRUD paketas pagreitina programuotojų darbą, palengvina kasdienę veiklą ir užtikrina pakartotinį naudojimą būsimuose projektuose. Tokiu būdu verslas gauna aukštos kokybės produktą per trumpesnę laiką.

Šiame darbe kuriamas CRUD funkcijų generavimo paketas populiariam ir plačiai naudojamam PHP programavimo kalbos karkasui Laravel. Pagrindinė paketo logika yra leisti naudotojui naudojant grafinę sąsają automatiškai sukurti modelį, vaizdo failą su atitinkamomis formomis ir valdiklio klasę su visais reikalingais metodais.

**Darbo tikslas** – sukurti CRUD funkcijų generavimo paketą Laravel karkasui, kuris būtų visiems viešai prieinamas per paketų tvarkyklę Composer. Šis paketas automatiškai sukuria grafinę sąsają, kurioje naudotojai gali generuoti migracijas, modelius ir valdiklius su visomis CRUD funkcijomis. Be to, duomenų validacija konfigūruojama naudojant grafinę sąsają.

### **Darbo uždaviniai:**

1. Išanalizuoti jau esamus Laravel karkaso CRUD funkcijų generavimo paketus, atrasti privalumus, trūkumus, netikslumus ir vietas, kurias galima tobulinti.

2. Suformuluoti kriterijus, kuriuos turi atitikti Laravel CRUD funkcijų generavimo paketas.
3. Teoriškai aprašyti visas paketo funkcijas ir galimybes, sistemos aktorius ir objektus bei suprojektuoti pagrindines UML diagramas.
4. Sukurti Laravel karkaso CRUD funkcijų generatorių ir įdėti jį į <https://packagist.org/> svetainę, kur jis būtų viešai prieinamas visiems.

Rašto darbo 1 skyriuje yra pateikiama CRUD generatorių analizė - aprašomi ir palyginami visiems viešai prieinami paketai, išryškinant kiekvieno privalumus bei trūkumus. Papildomai, yra pateikiamas galutinis sukurtų sprendimų analizės apibendrinimas. 2 skyriuje aprašomas teorinis sistemos modelis. Daugiausia dėmesio skiriama sistemos funkcionalumui, aprašomi funkciniai ir ne funkciniai reikalavimai, vaizduojamos sistemos pagrindinės UML diagramos - naudojimo atvejų, klasių, sekos, veiklos. Teoriniame sistemos modelyje papildomai pateikiama sistemos navigacijos diagrama bei duomenų bazės schema. 3 skyrius yra skirtas sistemos programinės realizacijos aprašymui. Pagrindė orientuojamasi sistemos architektūrą, naudojamas technologijas ir įrankius, išskiriami svarbiausi sistemos realizavimo sprendimai. Papildomai, detalai aprašomos visos pagrindinės klasės, nurodant jų atsakomybę, veikimo strategiją ir naudojimą sistemoje. Šio rašto darbo pabaigoje pateikiamos išvados, rekomendacijos bei ateities darbų planas.

# 1. Sukurtų sprendimų analizė

Šiame skyriuje aprašoma panašių paketų, generuojančių CRUD funkcijas, analizė. Pagrindinis kriterijus - paketai turi būti suderinami su PHP programavimo kalbos karkasu Laravel. Pagrindinė analizės idėja buvo išsiaiškinti, kokius sprendimus žmonės dažniau renka. Prieš pradėdant analizę buvo išskirti vertinimo kriterijai, kurie padėjo objektyviai įvertinti kiekvieną analizuojamą paketą. Vertinimo kriterijais atsižvelgiama į tokius veiksnius kaip: paketo funkcionalumas, dokumentacija, naudojimo sunkumas, optimizavimas, efektyvumas. Pagrindinis analizės tikslas buvo išsiaiškinti, kokie veiksniai yra svarbiausi naudotojams ir rasti tobulą balansą tarp visų anksčiau paminėtų veiksnių.

## 1.1. Plačiai naudojamų Laravel karkaso CRUD generavimo paketų analizė

Buvo rasta daug įvairių CRUD generavimo paketų Laravel karkasui. Žmonės siūlo daugybę skirtingų sprendimų, pradedant nuo labiausiai paprastų paketų, kurie gali generuoti tik kai kuriuos metodus, kurie gali būti naudojami vėliau kode, baigiant sudėtingais su didelėmis funkcionalumo galimybėmis. Iš pirmo žvilgsnio, paprasčiausius CRUD paketus galima naudoti labai lengvai, tačiau jų galimybės yra labai ribotos ir gali būti tinkamos tik kraštutinėse situacijose. Kita vertus, yra daugybė sudėtingų CRUD generavimo paketų su plačiu funkcionalumu, bet jie reikalauja gilaus kodo supratimo, kitaip kalbant, jie labiau tinka patyrusiems programuotojams, nes dokumentacija ne visada yra išsamiai ir lengvai suprantama. Norėdamas atlikti konkretesnę paketų analizę, iš viso esamų CRUD generavimo paketų sąrašo išsirinkau penkis, kuriuos aprašysiu detalai ir pateksiu subjektyvų vertinimą kiekvienam iš jų.

### 1.1.1. *infyomlabs/laravel-generator* paketas

Pirmasis CRUD generatoriaus paketas, kurį pasirinkau analizei, yra plačiai žinomas ir populiarus laravel bendruomenėje *infyomlabs/laravel-generator* [2]. Iš pirmo žvilgsnio sunku jį apibūdinti kaip paketą, mano nuomone, geriau būtų pavadinti jį kaip nepriklausomą įrankį, skirtą generuoti CRUD, nes jis suteikia daug daugiau funkcionalumo ir galimybių nei įprastas paketas. Verta paminėti, kad generatoriaus paketą sukūrė "InfyOm Technologies" įmonė, kuri taip pat teikia įvairias žiniatinklio kūrimo paslaugas. Nepaisant to, kad CRUD generatorius nėra komercinis produktas, mes matome labai profesionalų požiūrį į paketo kūrimą, priežiūrą ir dokumentavimą.

Paketas iš tikrųjų turi platų funkcionalumą ir gali padėti kasdieniame programuotojo darbe. Paketo dokumentacijoje galime rasti visą pagrindinių funkcijų sąrašą. Šiame sąrašė matome tokias funkcijas kaip: modelių, migracijų, vaizdų ir valdiklių generavimas, automatizuoti maršrutai ir testai. Papildomai paketas suteikia patogią grafinę sąsają, leidžiančią valdyti visus generatoriaus procesus. Valdymo skydelis apsaugotas autorizacija, kuri taip pat sukurama automatiškai. Automatiškai sugeneruoti modeliai, valdikliai, migracijos ir vaizdai matomi valdymo skydelyje, kur naudotojai gali juos redaguoti rankiniu būdu. Paketas palaiko tris įvesties metodus, kai kalbama apie CRUD generavimą, įvestį iš konsolės, failo ir lentelės. Apibendrinant paketo galimybes matome, kad paketas suteikia visas naudingas ir reikalingas funkcijas bei naudoja geriausius sprendimus, kurie gali padėti programuotojams ir net paprastiems darbuotojams.

Dar vienas paketo aspektas, kurį verta paminėti, yra dokumentacija. Dokumentacija rodoma atskirame, specialiai tam sukurtame interaktyviame interneto puslapyje. Dokumentacijoje galime rasti visų funkcijų detalius aprašymus, naudojimo instrukcijas ir net aiškius pavyzdinius failus.

Dokumentacija turi atskirus skyrius, kuriuose minimi visi techniniai reikalavimai ir priklausomybės, kurių reikalauja paketas. Papildomai įmonė parengė vaizdo įrašus apie paketą, kurie padeda naudotojams priprasti prie funkcionalumo ir parodo geriausius būdus, kaip juo naudotis. Apibendrinant, matau, kad šio paketo dokumentacija parengta rūpinantis jo naudotojais, ji yra išsami, informatyvi, o reikiama informacija yra lengvai randama, galiu teigti, kad tai puikus pavyzdys, kaip turi atrodyti dokumentacija.

InfyOm CRUD paketo analizės metu atkreipiau dėmesį į vieną iš svarbiausių aspektų – paketo diegimą. Paketas yra gana sudėtingas, turi platų funkcijų sąrašą ir reikalauja daug priklausomybių, todėl diegimas skiriasi nuo įprasto paketo diegimo. Galiu pasakyti, kad diegimo procesas yra komplikotas, tačiau ypač dėl šios priežasties dokumentacijoje yra atskiras skyrius, kuriame yra aprašomi visi veiksmai, kuriuos naudotojas turi atlikti, kad diegimas būtų sėkmingas. Dokumentacijoje pateikiami du generatoriaus diegimo būdai. Pirmasis variantas yra įdiegti naudojant specialiai paruoštus šablonus. Toks diegimo būdas yra gana paprastas, naudotojui tereikia pasirinkti iš 3 skirtingų šablonų tą, kuris geriausiai atitinka jo poreikius ir galima pradėti darbą, tačiau tai tinka tik tada, kai naudotojas pradeda naują projektą. Diegimas į esamus Laravel projektus yra sudėtingesnis, nes naudotojas turi pasirinkti visomis paketo priklausomybėmis ir jų versijomis. Nepaisant to, kad diegimas gali reikalauti, kad naudotojas atliktų dar kelis papildomus veiksmus, visi jie yra išsamiai dokumentuoti ir tai padeda bet kokiais probleminiais atvejais.

Prieš darydamas bendrą paketo išvadą, norėčiau paminėti visus privalumus ir trūkumus, kuriuos pastebėjau analizės ir praktinio naudojimo metu. Galiu tvirtai teigti, kad pagrindinis šio paketo privalumas yra platus funkcionalumas, apimantis beveik visą nekomplikuotą CRUD funkcijų generavimą. Dar vienas privalumas – generatoriaus techninė dalis. Visas šaltinio kodas išlaiko švarius kodo standartus ir rekomendacijas. Kadangi visas generatorius yra parašytas naudojant SOLID dizaino modelį, programuotojo požiūriu nėra sunku išsiaiškinti, kaip veikia tam tikros paketo funkcijos, be to paketas yra atviras išplėtimui. Paskutinis didelis privalumas – dokumentacija, apimanti visus svarbius dalykus ir padedanti naudotojams naudotis šiuo paketu. Kalbant apie trūkumus, pirmiausia norėčiau paminėti, kad paketas gali sukelti sunkumų nepatyrusiems programuotojams. Pagrindinis aspektas čia yra sudėtinga paketo struktūra, iš pirmo žvilgsnio sunku suprasti, kaip funkcijos generuoja CRUD, kaip turi atrodyti įvestis ir kokių pasiruošimo žingsnių reikia, todėl dokumentacijai teks skirti daug laiko. Dar vienas trūkumas yra generatoriaus funkcionalumo apribojimas. Nepaisant to, kad generatorius turi platų funkcijų sąrašą, dažniausiai šios funkcijos apima paprastą logiką turinčias funkcijas, o tai reiškia, kad gali būti sunku arba net neįmanoma sukurti sudėtingą CRUD, kuriam reikalinga neįprasta logika ir kuri laikosi savo taisyklių.

Apibendrinant galiu teigti, kad nepaisant visų paketo trūkumų, generatorius gali padėti daugelyje kasdinių situacijų. Norint geriau suprasti ir panaudoti visą paketo potencialą, rekomenduojama turėti tam tikros praktikos programuojant, be patirties iš pradžių gali būti sunku. Mano paties pastebėjimas yra tai, kad geriau naudoti šį generatorių, kai dirbate su nauju projektu, bet jei jums reikia CRUD generatoriaus esamam projektui, geriau pasirinkti kitą paketą.

### 1.1.2. *roland/crud* paketas

Kitas CRUD generatorius, kuris buvo pasirinktas analizei, yra *roland/crud* [6]. Šis paketas taip pat plačiai žinomas Laravel bendruomenėje, bet, deja, tai sena sistema, kuri buvo skirta Laravel 5.4.x versijai ir dabar yra nebepalaikoma. Nepaisant paminėtos priežasties, šis paketas analizuojamas su tikslu rasti privalumus ir trūkumus, išsiaiškinti, kokių sprendimų buvo imtasi.

Paketas gali generuoti migraciją ir valdiklio klasę su kūrimo, skaitymo, atnaujinimo ir ištry-

nimo metodais. Papildomai, generatorius gali sukurti vaizdo failą su įvesties duomenų formomis ir tinkamą kiekvieno metodo maršrutą. Šis paketas naudoja konsolę visoms komandoms ir parametrams įvesti. Pagrindinis šio generatoriaus privalumas yra paprastas diegimo procesas, nes jis neturi daug priklausomybių, kurias reikia įdiegti. Galiu pasakyti, kad tai yra labai nepriklausomas įrankis. Papildomai, dėl paprastos struktūros ir komandų, paketas yra atviras jaunesniems programuotojams, kurie neturi daug patirties. Pagrindinis šio paketo trūkumas, kurį pastebėjau, yra tai, kad generatoriaus procesas negali sukurti modulių ryšio. Tai reiškia, kad jei yra reikalingas ryšys tarp objektų, vienintelė galimybė yra juos sujungti rankiniu būdu, modifikuojant šaltinio kodą. Dar vienas paketo trūkumas yra tas, kad sugeneruotose formose gali būti tik *input* tipo laukai. Paaiškinsiu, ką čia turiu omenyje, jei naudotojas nori įrašyti tekstą, daug geriau ir patogiau įvesti tekstą i *textarea* tipo lauką. Paskutinis paketo trūkumas yra išsamios dokumentacijos trūkumas. Dabartinė dokumentacija yra labai trumpa ir aprašo diegimo procesą, kuris ir taip yra gana paprastas. Kalbant apie generatoriaus funkcionalumą, yra aprašytas tik vienas metodas be aiškių pavyzdžių, kas nėra labai patogu.

Apibendrinant reiktų pasakyti, kad šiuo metu negalima naudoti šio paketo, nes jis yra pasenęs ir nebepalaikomas, tačiau jis gali būti naudojamas kaip pavyzdys kuriant CRUD generatorius ateityje, dėl savo švarios ir paprastos struktūros.

### 1.1.3. *crestapps/laravel-code-generator* paketas

Kitas paketas, kuris buvo pasirinktas analizei, yra *crestapps/laravel-code-generator* [1]. Tai CRUD generatoriaus paketas, sukurtas maždaug prieš 4 metus, bet vis dar palaikomas iki šiol. Iš pirmo žvilgsnio akivaizdu, kad paketas buvo sukurtas rūpestingai, o dėmesys detalėms yra aukšto lygio. Papildomai yra prieinamą specialiai autoriaus parengta atskira svetainė su išsamia dokumentacija, kuri turi padėti naudotojams susipažinti su paketu. Paketas gali generuoti migracijas, modelius, valdiklius, vaizdo failus ir turi dar daugiau unikalių galimybių, tačiau visa tai bus išsamiai aprašyta sekančiose poskyriuose.

Kalbant apie šio paketo funkcijų sąrašą jis tikrai yra ilgas ir susideda iš įvairių naudingų funkcijų, kurios gali padėti programuotojams kasdieniame darbe. Analizės metu daugiausia dėmesio skyriau unikalioms funkcijoms ir sprendimams, kurių nėra kituose analizuojamuose paketuose. Viena iš naujų šio paketo galimybių, kurią pastebėjau, yra tai, kad jis gali dirbti ne tik su naujais duomenimis ir objektais, bet ir su senais duomenimis, kurie saugomi duomenų bazėje. Šis faktas žymiai padidina paketo naudojimą, kai kalbame apie jau esamus ar pradėtus projektus. Kita svarbi funkcija, kurią suteikia paketas, yra darbas su failų įkėlimu, kuris taip pat yra retas dalykas panašiose pakuotėse. Dar vienas įdomus dalykas, kurį pastebėjau, yra tai, kad paketas saugo sugeneruotus laukus JSON failuose. Toks sprendimas suteikia didelį lankstumą, nes pasikeitus verslo reikalavimams, tuos pakeitimus vis tiek bus lengva įvesti į sugeneruotas formas, nekeičiant šaltinio kodo. Verta paminėti, kad paketas turi išmaniąją duomenų validacijos funkciją. Tai reiškia, kad naudotojas, generuojant objektus, gali pasirinkti iš plataus validavimo taisyklių sąrašo tas, kurios bus taikomos tiek serverio, tiek kliento pusėje. Paskutinė svarbi savybė yra ta, kad paketas suteikia vaizdo failų pilną pritaikymą pagal poreikius. Tai reiškia, kad naudotojas turi galimybę keisti vaizdo failų šablonus, išvaizdą ir visus naudojamus stilius pagal savo poreikius. Visos aprašytos funkcijos yra tik maža dalis visų funkcijų, kurias gali turėti naudotojas, visas funkcijų sąrašas pateikiamas oficialioje paketo svetainėje.

Dokumentacija visada yra svarbi kiekvieno įrankio ar paketo dalis. Kalbant apie *crestapps* paketą, kaip buvo minėta pradžioje, autorius parengė internetinę svetainę, kurioje naudotojas gali

rasti išsamią dokumentaciją. Dokumentacija suskirstyta į skyrius, pradedant nuo bendriausių, kaip įvadas ir paketo funkcijų sąrašas, ir baigiant išplėstiniais skyriais su sudėtingiausių paketo funkcijų aprašymais. Dokumentacijos skyriuose naudojamos aiškos antraštės, kurios leidžia naudotojui lengvai rasti tai, ko jis ieško. Papildomai norėčiau paminėti, kad yra paruošti vaizdo įrašai, kurie padeda naudotojui susipažinti su paketu. Baigdamas paketo dokumentacijos skyrių norėčiau pasakyti, kad tai taip pat geras pavyzdys to, kaip turi būti parengta dokumentacija.

Paminėjus visus svarbius dalykus, laikas apibūdinti pagrindinius paketo privalumus ir trūkumus. Pirmasis pagrindinis šio paketo privalumas yra platus funkcijų sąrašas. Paketas gali padėti programuotojams ir automatizuoti daugelį procesų, kurie gali užtrukti daug laiko. Kitas privalumas yra tas, kad paketas gali dirbti su esamais ar pradėtais projektais, o juo labiau turi būtina funkcionalumą dirbti su senais duomenimis. Paketo diegimo procesas yra gana paprastas ir greitas, norint įdiegti šį paketą nereikia programavimo patirties, todėl diegimo procesas taip pat gali būti laikomas pranašumu. Svarbiausias privalumas yra išsami pakuotės dokumentacija. Kaip jau buvo minėta anksčiau, dokumentacija apima visus paketo funkcijų aprašymus su praktiniais pavyzdžiais ir net vaizdo medžiaga. Paskutinis pranašumas yra tas, kad paketas generuoja daugkartinį, švarų ir lengvai suprantamą bei modifikuojamą šaltinio kodą. Kalbant apie trūkumus, jų nėra daug, bet vis tiek paminėsiu tuos, kuriuos pastebėjau. Sudėtinga paketo struktūra kartu su daugybe komplikotų funkcijų sukuria vaizdą, kuris gali išgąsdinti programuotojus, neturinčius didelės patirties. Tai reiškia, kad paketas nėra visiškai atidarytas jaunesniems programuotojams, jie dažniau norės praleisti šį paketą, nei jį perprasti. Antras trūkumas – paketas neturi grafinės sąsajos, visoms funkcijoms naudoja konsolinį terminalą. Galima gana lengva paaiškinti, kodėl šiuo atveju tai yra trūkumas. Paketas turi daug sudėtingų funkcijų, kurioms reikalingi parametrai, o reikalingų parametrų skaičius gali būti didelis. Visi reikalingi parametrai turi būti įtraukti į vieną konsolės komandą, kuri galiausiai įgauna sudėtingą komandos sintaksę ir tokios komandos ilgis gali užtrukti net keletą kodo eilučių. Papildomai, tokias sintaksės įsiminti gana sunku, o tikimybė suklysti yra gana didelė.

Apibendrinant galima pasakyti, kad nepaisant visų paketo trūkumų, tai labai stiprus procesų automatizavimo įrankis. Pagrindinės paketo CRUD generavimo funkcijos yra sukurtos labai gerai ir pilnai atlieka savo darbą. Tačiau dėl sudėtingumo sunku suprasti, kaip naudoti tam tikras papildomas funkcijas. Norint išnaudoti visas šio paketo galimybes, rekomenduočiau jį tik programuotojams, turintiems didelę patirtį, o tiems, kuriems dar trūksta programavimo patirties, geriau pirmiausia pasidomėti paprastesniais paketais.

#### **1.1.4. *larafast/fastapi* paketas**

Paskutinytis analizuojamas Laravel CRUD generavimo paketas - *larafast/fastapi* [3]. Tai paketas, sukurtas 2020 m. ir palaikomas iki šių dienų. Paskutinė paketo versija yra suderinama su php 8.x ir Laravel 9.x versijomis. Šio paketo siūlomas generatorius šiek tiek skiriasi nuo visų anksčiau aprašytų. Nepaisant visų nedidelių skirtumų generatoriaus architektūroje, pagrindinis skirtumas yra tas, kad paketo generatorius yra skirtas generuoti CRUD API funkcionalumą. Paketo generatorius gali sukurti modelį, migraciją, išteklių modelį, API užklauso klasę, gamyklinį modelį ir valdiklį su CRUD funkcijomis tik su viena konsolės komanda.

Prieš darydamas išvadas, pateiksiu pagrindinius generatoriaus privalumus ir trūkumus, kurie buvo pastebėti analizės metu. Pagrindinis paketo privalumas, kuris pastebimas iš pirmo žvilgsnio, yra jo paprastumas. Tik su viena konsolės komanda naudotojas gali naudotis visomis siūlomomis paketo funkcijomis. Dar vienas privalumas – greitas diegimo procesas. Nepaisant to, kad gene-

ratorius priklauso nuo vieno papildomo paketo, diegimas vyksta su vienos komandos dėka. Verta paminėti, kad paketo autorius parengė trumpą, bet aiškią dokumentaciją. Dokumentacijoje aprašomas diegimo procesas ir pateikiami paketo generatoriaus funkcijų naudojimo pavyzdžiai. Kalbant apie paketo trūkumus, pirmasis ir svarbiausias paketo trūkumas yra tai, kad valdiklio klasė neturi jokių validacijos taisyklių. Pasak paketo autoriaus, visi laukai pagal nutylėjimą yra nustatyti kaip privalomi, tačiau jei naudotojui reikia daugiau validacijos taisyklių, jis gali savarankiškai jas pridėti į valdiklio klasę. Dar vienas pastebėtas paketo trūkumas yra tai, kad generatorius negali automatiškai kurti sąryšių tarp modelių.

Apibendrinant galima pasakyti, kad paketas yra vertas kiekvieno programuotojo dėmesio dėl savo paprastumo ir aiškumo. Generatoriaus siūlomas funkcionalumas gali ženkliai padėti kiekvienam programuotojui kasdienėje darbo rutinoje ir pagerinti darbo efektyvumą. Nepaisant to, kad paketas vis dar pasigenda kai kurių svarbių funkcijų, jis yra atviras ir kiekvienas naudotojas gali jį keisti pagal savo poreikius. Tai, kad šis paketas yra puikus pasirinkimas tiek esamiems, tiek naujiems projektams, parodo paketo atsisiuntimo statistiką, kurioje yra daugiau nei tūkstantis atsisiuntimų.

## **1.2. Sukurtų sprendimų analizės apibendrinimas**

Sukurtų sprendimų analizė parodė, kad visi analizuoti CRUD generatorių paketai yra labai skirtingi ir naudoja savo strategijas bei sprendimus. Visi šie paketai suteikia puikias galimybes optimizuoti programuotojo darbo rutiną, tačiau norint išnaudoti visą tų paketų potencialą labai svarbu kokių atveju jie bus naudojami. Pirmasis paketas - infyomlabs/laravel-generator geriausiai pasirodo naujuose projektuose kai juo naudojasi patyrę programuotojai. Pagrindinis trūkumas yra sudėtinga paketo sąsaja, kadangi ne visada aišku kaip turi atrodyti įvestis ir kokių pasiruošimo žingsnių reikia. Kalbant apie roland/crud paketą, jis yra pasenęs ir nebepalaikomas. Nepaisant to, kad praktiškai jo beveik neįmanoma naudoti, su jo vis tiek galima susipažinti norint įgyti žinių ir suprasti, kaip kuriami CRUD generatorių paketai. Trečiasis paketas - crestapps/laravel-code-generator tai yra galingas įrankis generuoti CRUD, kuris parodo visą potencialą, kai yra naudojamas pradėtuose ir senuose projektuose dėl savo galimybės dirbti su senais duomenimis. Kaip pagrindinį trūkumą galima paminėti, kad paketo perpratimas reikalauja daug laiko ir programavimo patirties. Paskutinis išanalizuotas paketas - larafast/fastapi yra puikus paketo, turinčio konkrečią specializaciją, pavyzdys. Paketo funkcionalumas suteikia galimybę generuoti CRUD API, todėl tinka tiek naujiems, tiek seniems projektams, kurių struktūra numato API naudojimą. Pagrindinis paketo trūkumas yra tai, kad sugeneruotos valdiklių klasės neturi validacijos taisyklių.

## 2. Sistemos modelis

Šiame skyriuje aprašomas CRUD generatoriaus paketo funkcionalumas, dizainas. Minima apie pagrindinius aspektus ir sprendimus projektuojant sistemą. Taip pat pateikiami funkciniai ir nefunkciniai reikalavimai, pagrindinių funkcijų UML diagramos - naudojimo atvejų, veiklos, klasių, sekos. Taip pat pavaizduota ir aprašyta duomenų bazės schema ir sistemos navigacijos modelis.

### 2.1. Sistemos funkcionalumas

Paketo pagrindinė užduotis yra suteikti galimybę žmonėms, neturintiems programavimo patirties, automatiškai generuoti CRUD funkcijas su migracijom ir vaizdo failais, nekeičiant šaltinio kodo. Svarbiausia sukurti patogią grafinę sąsają, kuri būtų minimalistinio dizaino, bet suteiktų visas reikalingas funkcijas ir galimybes naudotojams. Grafinė sąsaja turi leisti naudotojui matyti visas sugeneruotas migracijas, modelius, peržiūrėti failus ir valdiklius. Verta paminėti, kad visi vaizdo failai turi palaikyti konfigūruojamą puslapiavimą. Papildomai, paketas privalo turėti galimybę dirbti su objektų ryšiais vienas su daugeliu ir daug su daugeliu. Paskutinis svarbus aspektas yra tai, kad generatorius turi žinoti, kaip sukurti tinkamą duomenų validaciją, kai naudotojas to reikalauja. Paketas neturės autorizacijos ar kitų saugumo priemonių. Toks sprendimas sumažins paketo priklausomybių kiekį, supaprastins diegimo procesą ir leis išvengti konfliktų, galinčių kilti, kai žmonės bandys įdiegti paketą į projektą su jau veikiančiu autorizavimu. Siekiant suteikti galimybę plėsti paketo funkcionalumą, visi maršrutai bus konfigūruojami, tai bus naudinga, pavyzdžiui, jei naudotojas norės juos apsaugoti autorizacija.

#### 2.1.1. Funkciniai reikalavimai

Sistemai keliami tokie funkciniai reikalavimai:

- Paketas turi būti pritaikytas Laravel karkaso 9.x versijai.
- Paketas privalo turėti grafinę sąsają CRUD generavimui.
- Paketo grafinės sąsajos dizainas turi būti minimalistinis ir patogus naudotojui.
- Paketo grafinė sąsaja privalo turėti galimybę rodyti visus sugeneruotus objektus.
- Paketo diegimo procesas turi būti paprastas ir greitas.
- Paketas turi būti nepriklausomas, nereikalaujantis išorinių priklausomybių.
- Generatorius turi gebėti kurti migraciją.
- Generatorius turi gebėti kurti modelį.
- Generatorius turi gebėti kurti vaizdo failą su įvesties formomis.
- Generatorius turi gebėti kurti valdiklį su kūrimo, skaitymo, atnaujinimo ir trynimo metodais.
- Generatorius turi sugebėti sukurti tinkamą duomenų validaciją.
- Paketo grafinė sąsaja turi leisti naudotojui nustatyti norimus sąryšius tarp objektų.
- Paketas turi suteikti galimybę konfigūruoti maršrutus.

### 2.1.2. Nefunkciniai reikalavimai

Sistemai keliami tokie nefunkciniai reikalavimai:

- Paketas turi būti saugus.
- Paketas turi būti atvirojo kodo.
- Paketas privalo turėti išsamią dokumentaciją.
- Paketas privalo būti laisvai prieinamas visiems.
- Generatorius turi veikti greitai, be gedimų ar trikdžių.
- Paketo šaltinio kodas turi atitikti PSR-12 reikalavimus.
- Paketo šaltinio kodas turi atitikti SOLID projektavimo šabloną.

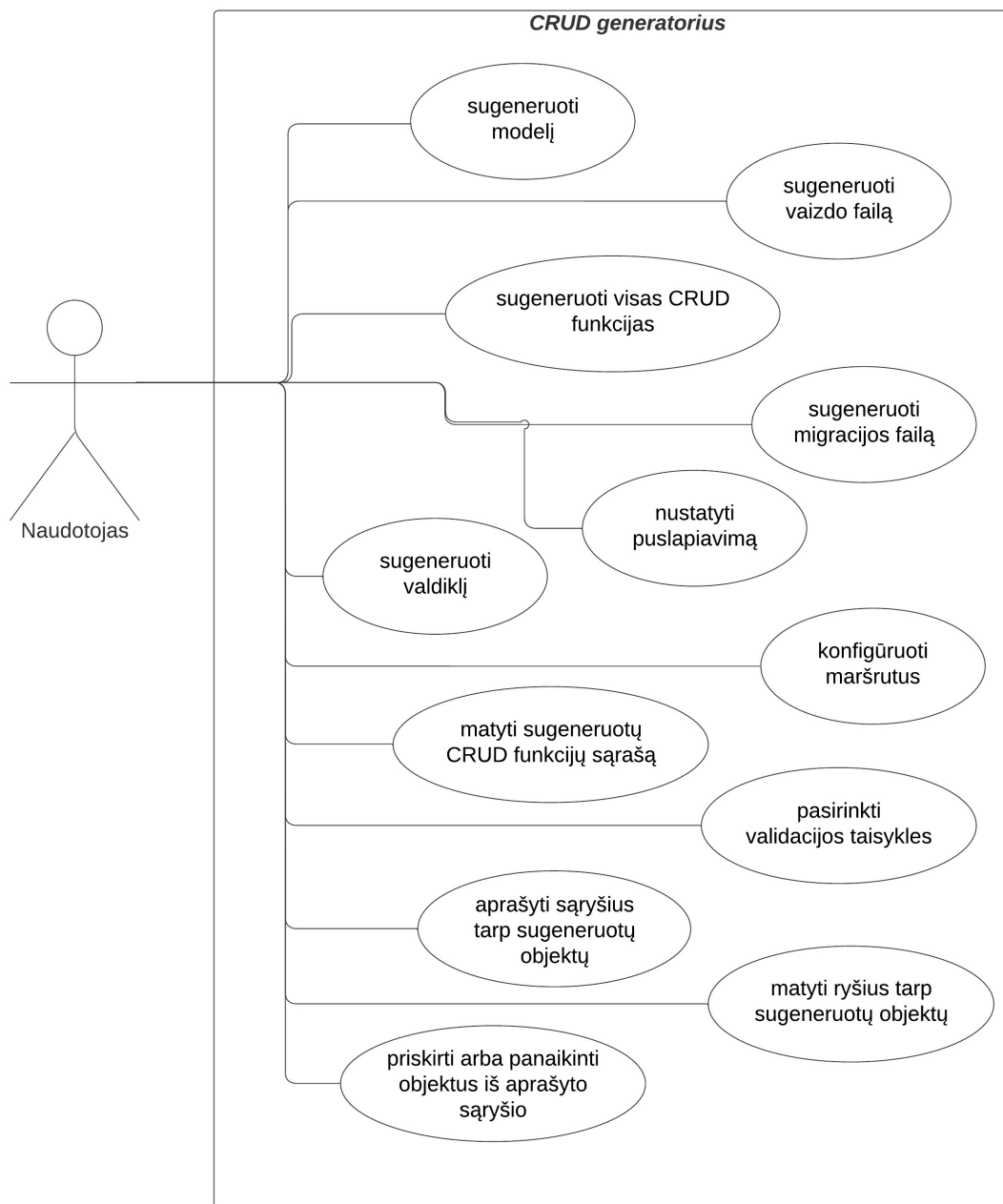
## 2.2. UML diagramos

Šiame poskyryje pateiktos ir aprašytos įvairios diagramos: UML panaudos atvejų diagrama, UML klasių diagrama, UML sekos diagrama, UML veiklos diagrama, sistemos navigacijos diagrama ir duomenų bazės schema.

### 2.2.1. Naudojimo atvejų diagrama

Visos CRUD generatoriaus paketo funkcijos rodomos panaudos atvejų diagramoje 1 pav. Šios funkcijos yra pasiekiamos naudotojui sėkmingai įdiegus paketą. Sistemoje yra vienas aktorius - naudotojas. Jis gali atlikti tokias funkcijas:

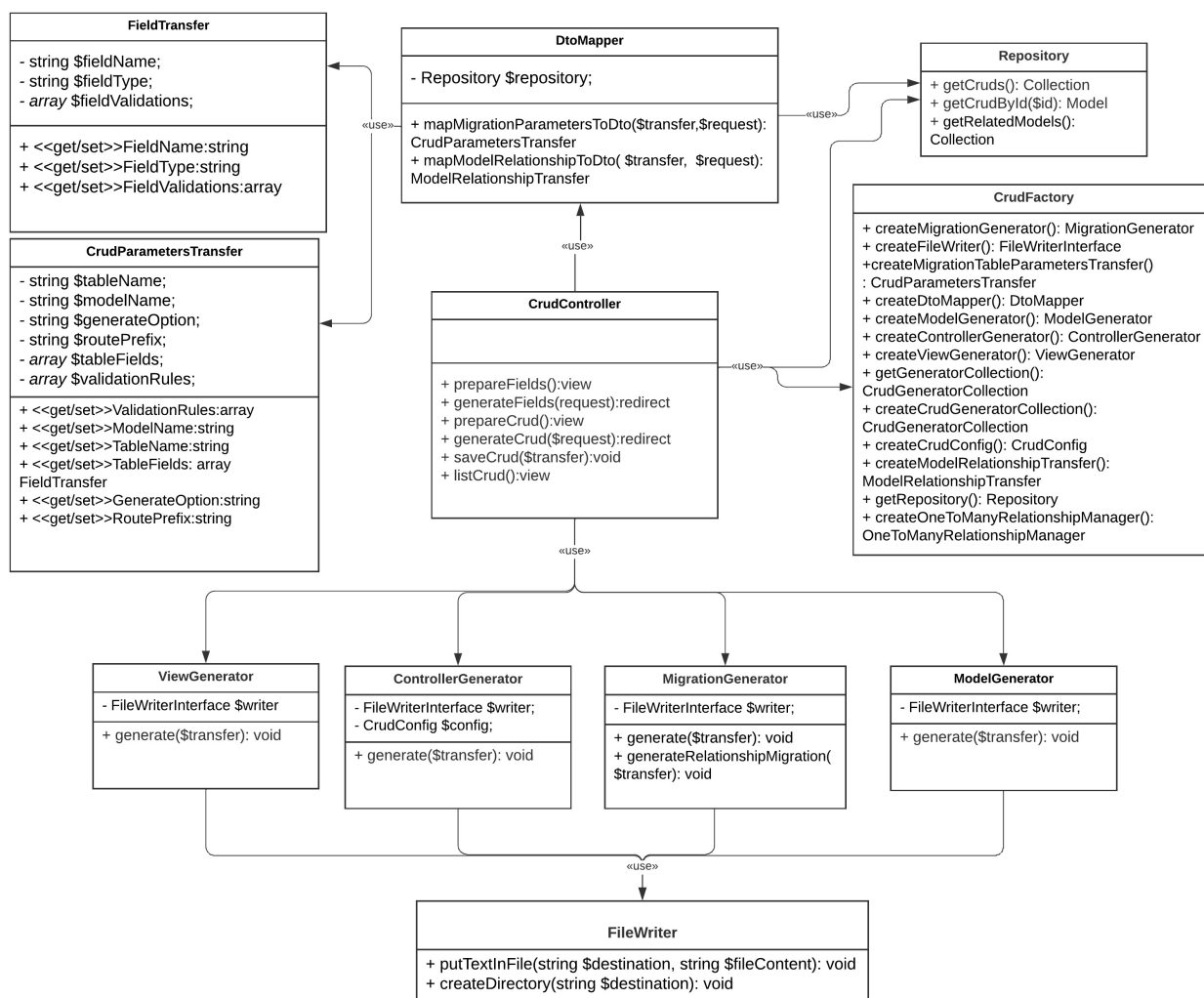
- Užpildžius visus reikiamus duomenis ir validacijos taisykles į generavimo formą, automatiškai sugeneruoti visą CRUD su migracija, modeliu, valdikliu ir vaizdo failu.
- Užpildžius visus reikiamus duomenis ir pasirinkus tik migracijos generavimą, automatiškai sugeneruoti migraciją.
- Užpildžius visus reikiamus duomenis ir pasirinkus tik modelio generavimą, automatiškai sugeneruoti modelį.
- Užpildžius visus reikiamus duomenis ir pasirinkus tik valdiklio generavimą, automatiškai sugeneruoti valdiklį.
- Užpildžius visus reikiamus duomenis ir pasirinkus tik vaizdo failo generavimą, automatiškai sugeneruoti vaizdo failą.
- naudotojas gali pasirinkti kiekvieno lauko validacijos taisykles.
- naudotojas gali matyti visą pilnai sugeneruotų CRUD sąrašą.
- naudotojas gali konfigūruoti automatiškai generuojamų CRUD objektų maršrutų priešdėlius.
- naudotojas gali aprašyti sąryšius tarp sugeneruotų objektų.
- naudotojas gali matyti visus aprašytus ryšius tarp sugeneruotų objektų.
- naudotojas gali priskirti arba panaikinti objektus iš aprašyto sąryšio.
- naudotojas gali konfigūruoti puslapiavimą.



1 pav. Panaudos atvejų diagrama

### 2.2.2. Klasių diagrama

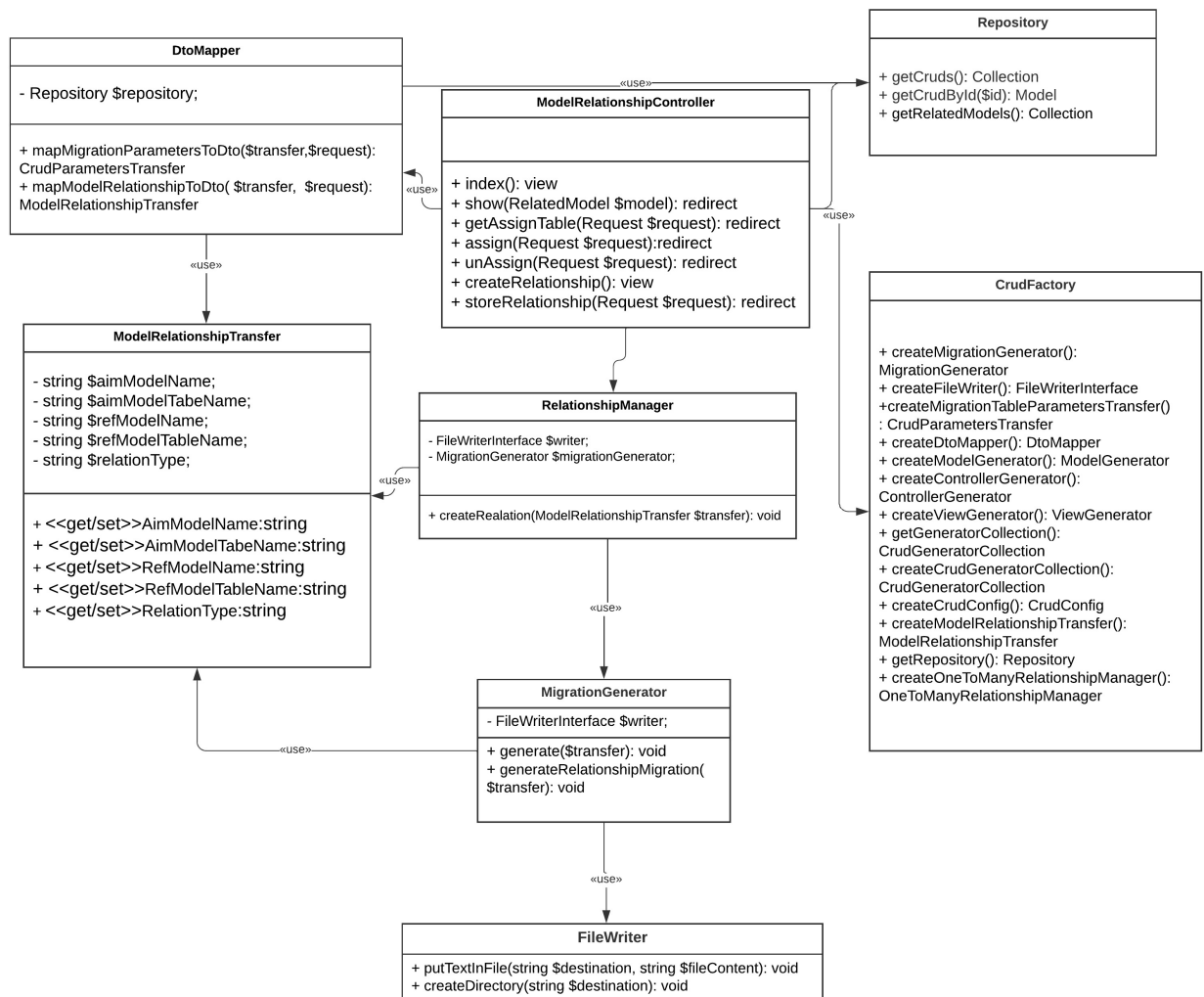
CRUD funkcijų generavimo klasių diagrama vaizduojama 2 pav. Funkcija prasideda nuo pagrindinės valdiklio klasės „CrudController“. Šis valdiklis yra pagrindinis deleguotojas, kuris valdo visą srautą, iškviestas kitas svarbias klases. Pagrindinės klasės, kurias naudoja valdiklis, yra šios: „CrudFactory“ klasė, kuri inicijuoja visus objektus, „Repository“ klasė, nuskaitanti duomenis iš duomenų bazės, „DtoMapper“ klasė, paverčianti visus duomenis į naudingesnę struktūrą, ir keturi pagrindiniai generatoriai, generuojantys CRUD failus. Visa tai, kas paminėta aukščiau, leidžia teigti, kad „CrudController“ klasė negali veikti atskirai ar savarankiškai, ji yra priklausoma nuo kitų klasių. Kalbant apie generatorių klases, jos taip pat nėra nepriklausomos, nes visos jos naudoja „FileWriter“ klasę, kuri iš tikrųjų sukuria naujus failus ir įrašo į jas atitinkamą turinį. Verta paminėti, kad toks klasių skirstymas į mažesnes dalis ir atsakomybės atskyrimas sukuria daug priklausomybių visoms klasėms, tačiau žymiai pagerina programos šaltinio kodo skaitomumą ir užtikrina funkcijų pakartotinį naudojimą.



2 pav. CRUD generavimo klasių diagrama

Kalbant apie sąryšių priskyrimo funkciją (žr. 3 pav.), ji turi labai panašią architektūrą ir naudojamus sprendimus kaip CRUD generavimo funkcija, kuri buvo aprašyta anksčiau. Funkcija pradeda veikti nuo „ModelRelationshipController“ valdiklio. Ši klasė atsakinga už įvesties duomenų rinkimą ir tai yra įėjimo taškas iš maršrutų, kurie yra susiję su modelio santykių priskyrimu.

Valdiklis naudoja „DtoMapper“, kuris transformuoja duomenis į naudingesnę duomenų struktūrą. „RelationshipManager“ klasė yra atsakinga už naujų sąryšių tarp modelių kūrimą. Ši klasė išsaugo tuos ryšius kaip įrašus duomenų bazėje ir iškviečia klasę „MigrationGenerator“, kuri generuoja atskirą migraciją su aprašytu ryšiu.

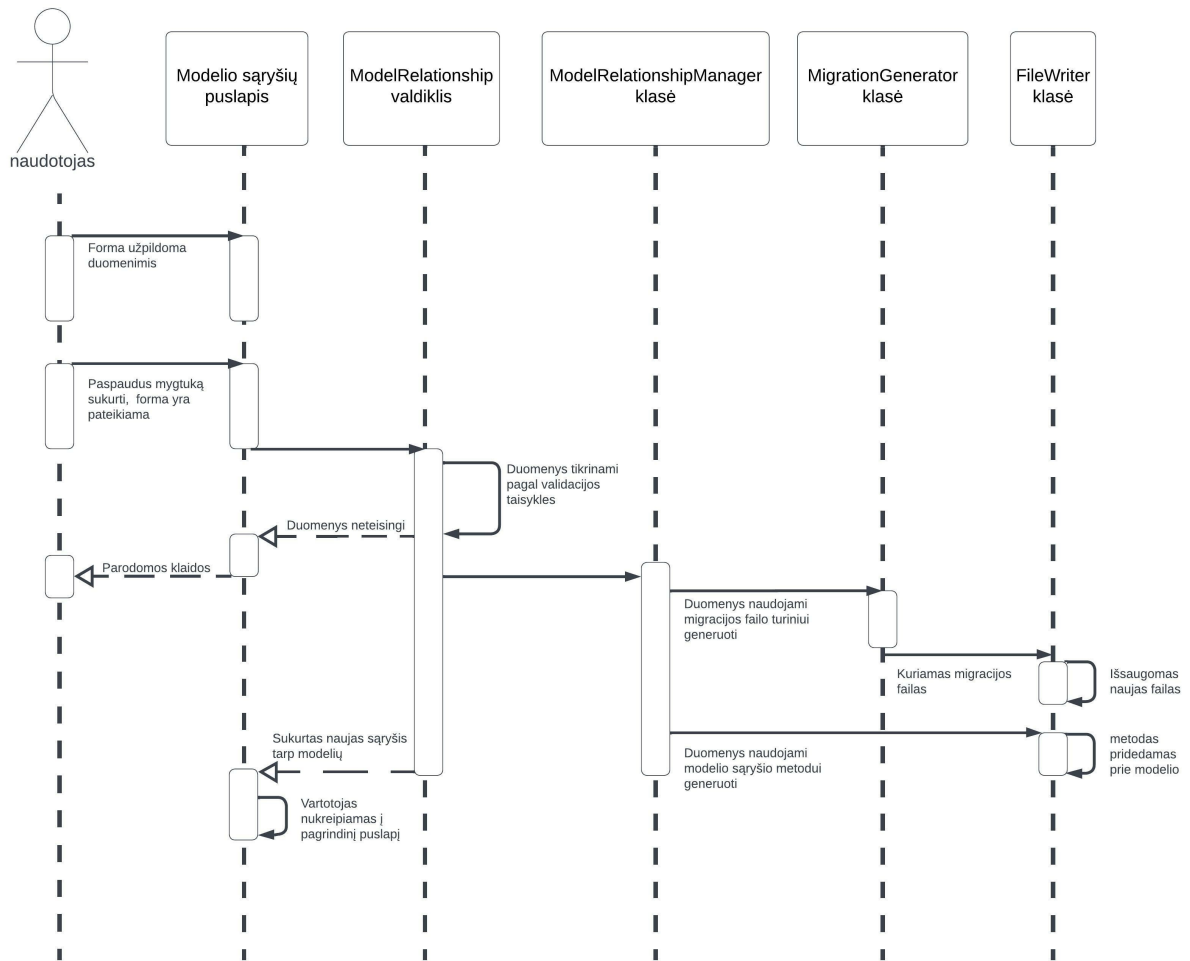


3 pav. Sąryšio priskyrimo klasių diagrama

### 2.2.3. Sekos diagrama

UML sekos diagrama pateikiama 4 pav. Joje pavaizduota naujo ryšio tarp sugeneruotų modelių kūrimo funkciją. Kaip parodyta diagramoje, funkcija prasideda pagrindiniame sąryšių kūrimo puslapyje, kuriame naudotojas turi pateikti formą su pasirinktais duomenimis apie modelių pavadinimus ir sąryšio tipą. Pateikta forma su duomenimis yra tikrinama „ModelRelationshipController“ valdiklyje, pasirodžius klaidoms naudotojas nukreipiamas atgal į pagrindinį puslapį su klaidos pranešimu. Jei pateiktos formos validacija praeina sėkmingai, formos duomenys perduodami iš valdiklio į „RelationshipManager“ klasę, kuri yra atsakinga už naujo ryšio sukūrimą. Tam, kad sukurti naują ryšį tarp sugeneruotų modelių, „RelationshipManager“ klasė turi atlikti du veiksmus. Pirmas žingsnis yra sugeneruoti migracijos failą, kuris pridės reikiamus laukus modeliams, kurie bus susiję sąryšių. Šiam veiksmui naudojama „MigrationGenerator“ klasė, kuri generuoja reikiamus laukus, ir „FileWriter“ klasė, kuri faktiškai sukuria migracijos failą. Antrasis žingsnis, kurį

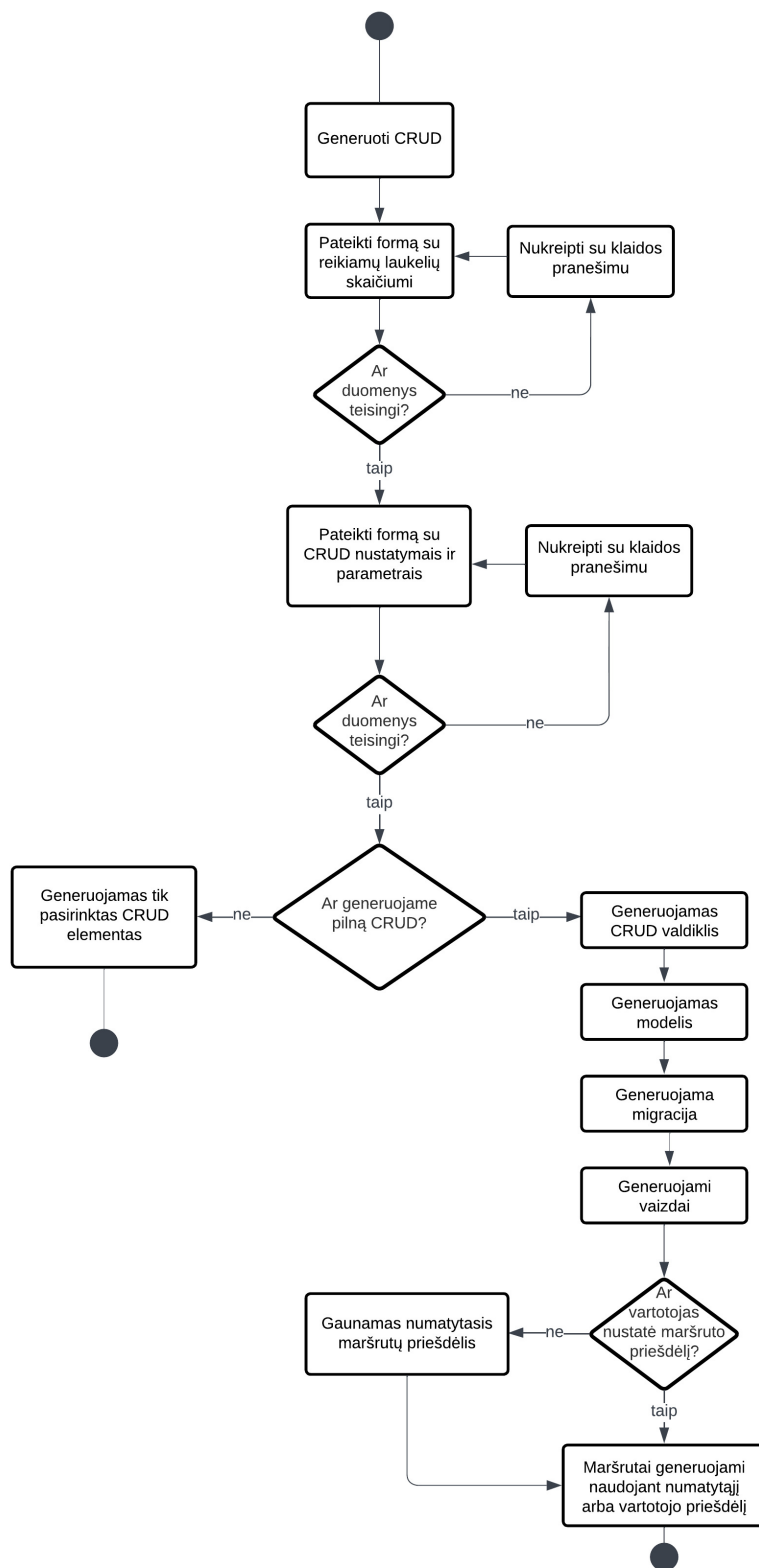
reikia padaryti, yra pridėti sąryšių metodus susijusiems modeliams. Tai atliekama „FileWriter“ klaseje, kuri gali rasti susijusių modelių failus ir pridėti reikiamus metodus.



4 pav. Modelio sąryšio kūrimo sekos diagrama

## 2.2.4. Veikos diagrama

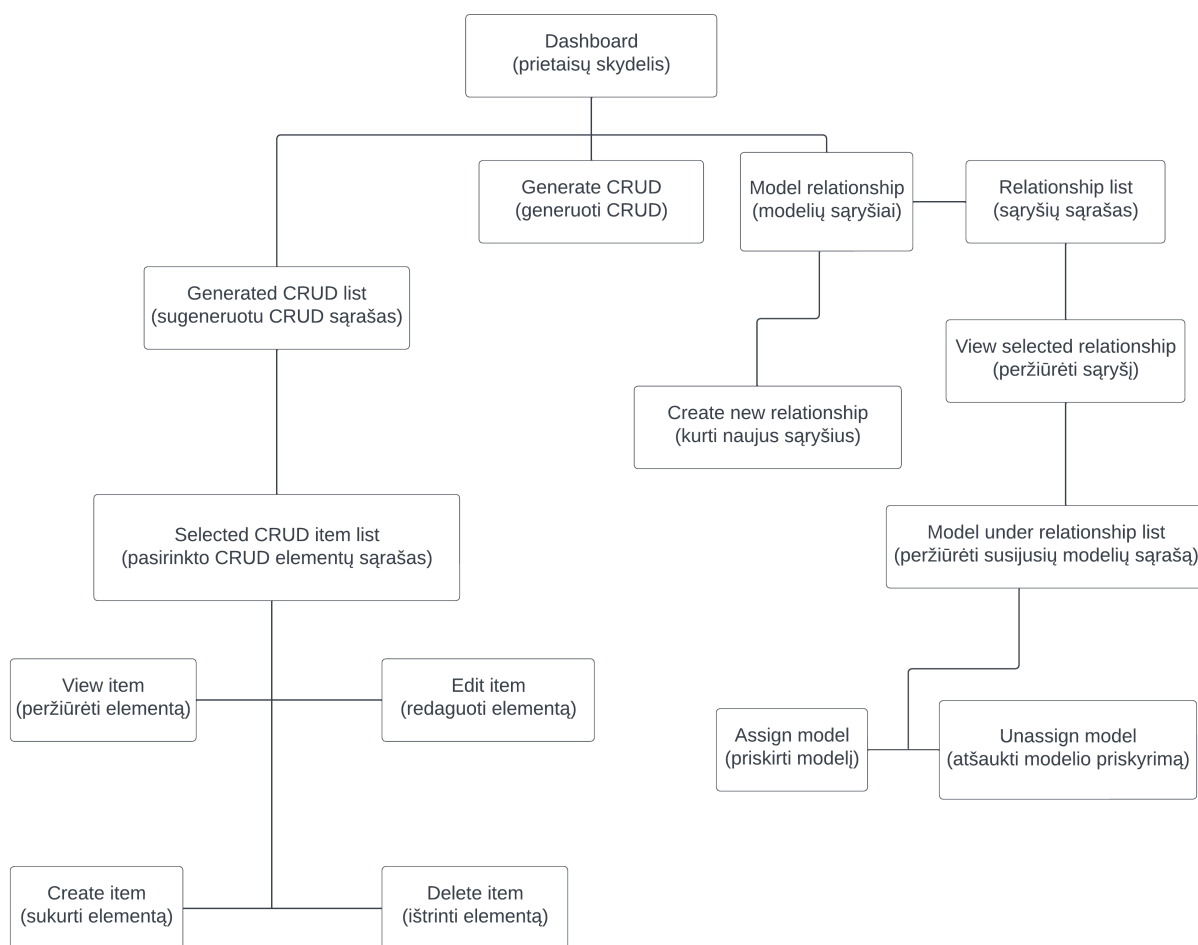
UML veiklos diagrama pateikiama 5 pav. Joje pavaizduotas CRUD generavimo funkcijos veiksmų eiliškumas. Kai naudotojas paspaudžia mygtuką generuoti CRUD, programa grąžina formą, kurioje naudotojas turi užpildyti, kiek laukų turi būti sugeneruota būsimam modeliui. Pateikta forma bus patikrinta valdiklyje, jei atsiras klaida, naudotojas bus nukreiptas atgal į pradinį puslapį su klaidos pranešimu, kitas atvejis naudotojas gaus kitą formą, kurioje galės užpildyti jam reikalingus CRUD nustatymus. Panašiai kaip ir ankstesniame žingsnyje, forma taip pat tikrinama valdiklyje. Kitas valdiklio veiksmas yra patikrinti, koks CRUD elementas turi būti sugeneruotas. Jei naudotojas pasirenka generuoti tik migraciją, vaizdo failą, valdiklį ar modelį, valdiklis iškvies generatoriaus klasę ir sugeneruos tik pasirinktą elementą. Kai naudotojas pasirenka sugeneruoti pilną CRUD, iškviečiami visi keturi generatoriai, kurie generuoja migracijas, modelius, valdiklius ir vaizdo failus. Kai visas CRUD yra jau sugeneruotas, belieka tik sukurti tinkamus maršrutus. Šiame žingsnyje valdiklis patikrina, ar naudotojas nustatė maršruto priešdėlį. Jei priešdėlis nenustatytas, valdiklis paima numatytąjį priešdėlį iš konfigūracijos klasės. Galiausiai valdiklis sukuria naujus CRUD funkcijų maršrutus.



5 pav. Veiklos diagrama

### 2.2.5. Sistemos navigacijos diagrama

UML paketo sistemos navigacijos diagrama pateikiama 6 pav. Ši diagrama (žr. 6 pav.) visiškai atspindi tikrąją grafinę sąsają, kurią pateikia paketas. Pagrindiniame puslapyje, kuris vadinamas prietaisų skydeliu, naudotojas gali pereiti į CRUD generavimo puslapį, modelio sąryšių puslapį, bei pamatyti visą sugeneruotų CRUD sąrašą. Iš sugeneruotų CRUD sąrašo naudotojas gali pasirinkti vieną ir pamatyti to jo elementų sąrašą. Papildomai, naudotojas su tais elementais gali atlikti visas CRUD funkcijas, tai reiškia sukurti elementą, peržiūrėti elementą, atnaujinti elementą ir ištrinti tą elementą. Modelio sąryšių puslapyje naudotojas gali sukurti naują sąryšį tarp sugeneruotų modelių arba pamatyti sukurtų sąryšių sąrašą. Šiame sąraše naudotojas gali pasirinkti sąryšį ir pamatyti, kurie modeliai yra įtraukti. Be to, naudotojas gali pasirinkti modelio objektą ir priskirti arba atšaukti sąryšiu susijusio modelio objektus.



6 pav. Sistemos navigacijos diagrama

### 2.2.6. Duomenų bazės schema

CRUD generatorius diegimo proceso metu duomenų bazėje sukuria dvi lenteles, kurias naudoja paketo funkcijos. Šios lentelės yra atvaizduotos 7 pav. ir aprašytos šiame skyriuje. Duomenų bazių lentelės, kuriose yra saugoma informacija apie sugeneruotų CRUD elementų įrašus, sukuriamos dinamiškai, todėl jos nerodomos 7 pav. Pagrindinės dvi lentelės, reikalingos sėkmingam paketo

darbui, tai „cruds“ lentelė ir „related\_models“ lentelė. „Cruds“ lentelė naudojama duomenims apie sugeneruotą CRUD saugoti, lentelę sudaro šie laukai:

- table\_name - saugo sugeneruotą lentelės pavadinimą.
- model\_name - saugo sugeneruotą modelio pavadinimą.
- route - saugo sugeneruoto CRUD pagrindinį maršrutą.
- created\_at - saugo datą, kada buvo sukurtas įrašas.
- updated\_at - saugo datą, kada buvo pakeistas įrašas.

„Related\_models“ lentelė saugo duomenis apie sukurtus sąryšius tarp sugeneruotų CRUD objektų, lentelę sudaro šie laukai:

- aim\_model - saugo pasirinkto modelio pavadinimą.
- ref\_model - saugo susijusio modelio pavadinimą.
- relation\_type - saugo sąryšio tipą.
- created\_at - saugo datą, kada buvo sukurtas įrašas.
- updated\_at - saugo datą, kada buvo pakeistas įrašas.

| rkliab_crud related_models   | rkliab_crud cruds         |
|------------------------------|---------------------------|
| id : bigint(20) unsigned     | id : bigint(20) unsigned  |
| aim_model : varchar(255)     | table_name : varchar(255) |
| ref_model : varchar(255)     | model_name : varchar(255) |
| relation_type : varchar(255) | route : varchar(255)      |
| created_at : timestamp       | created_at : timestamp    |
| updated_at : timestamp       | updated_at : timestamp    |

7 pav. Duomenų bazės schema

### 3. Sistemos programinė realizacija

Paketas suteikia galimybę automatiškai generuoti CRUD funkcijas su tinkamomis duomenų tikrinimo taisyklėmis ir objektų ryšiais. Visas šis funkcionalumas pasiekiamas daugelio klasių bendradarbiavimu ir įvairiais sprendimais, dėl kurių paketas veikia tinkamai. Be to, dėl išmaniųjų sprendimų, kurie buvo naudojami kuriant sistemą, šaltinio kodas yra gerai optimizuotas ir lankstus, yra atviras plėtrai ir atitinka visas SOLID rekomendacijas. Šiame skyriuje bus aprašytos naudojamos technologijos ir programavimo įrankiai, visos pagrindinės generatorių klasės ir sprendimai, turintys reikšmingą įtaką galutinei sistemai. Be to, paketas turi būti laisvai prieinamas visiems, todėl bus aprašytas ir visas paketo publikavimo į Packagist platformą procesas.

#### 3.1. Naudojami įrankiai ir technologijos

Pagrindinė technologija, kuri naudojama CRUD generatoriaus pakete, yra PHP 8.0 versijos programavimo kalba. PHP programavimo kalba yra privalomas pasirinkimas, nes paketas turi būti suderinamas su Laravel sistema, kuri taip pat sukurta naudojant PHP. Kita vertus, šis sprendimas leidžia naudoti įmontuotą Laravel funkcionalumą ir jos bibliotekas, kurios padeda dirbti su duomenų bazėmis ir failais. Viena iš tų svarbių bibliotekų yra Eloquent ORM duomenų bazės abstrakcija. Eloquent palengvina darbą su duomenų bazių lentelėmis, duomenų įrašų įterpimu, atnaujinimu ir ištrynimu. Duomenų bazė MariaDB buvo pasirinkta kaip pagrindinė saugykla sistemos duomenims saugoti dėl puikaus variklio veikimo greičio ir atvirojo kodo struktūros. Taip pat, MariaDb yra populiariausias pasirinkimas daugelyje projektų dėl tobulo suderinamumo su PHP programavimo kalba. Sistemos grafinei sąsajai naudojama HTML žymėjimo kalba kartu su CSS stiliaus aprašymo kalbos karkasu Bootstrap ir jo populiariu atvirojo kodo šablonu AdminLTE. Bootstrap karkaso sukurtos klasės yra plačiai naudojamos generuojamuose vaizdo failuose ir įvairiose formose, kurios renka duomenis CRUD generavimui. AdminLTE šablonas sukuria modernią grafinę sąsają su minimalistiniu, bet patogiu dizainu, kurią galima lengvai išplėsti.

#### 3.2. Struktūros aprašymas

Laravel karkasas visas priklausomybes ir bibliotekas saugo „vendor“ kataloge, tačiau paketų kūrimui rekomenduojama sukurti naują katalogą pavadinimu „packages“, kuriame bus saugomi visi paketo failai. Sukūrus paketo aplanko struktūrą, reikėjo atlikti du veiksmus, kurie leis Laravel matyti ir dirbti kartu su paketo failais. Pirmas žingsnis buvo aprašyti paketą composer.json faile, šis veiksmas leidžia Laravel pastebėti naują paketą ir jo failus. Antrasis žingsnis buvo sukurti paketo paslaugos teikėjo „serviceProvider.php“ failą ir prijungti jį prie Laravel. Šis failas yra atsakingas už paketo specifinę konfigūraciją, maršrutus, migracijas ir paketo vaizdo failus. Laravel karkasas turi specifinį konfigūracijos failą „app.php“, kuriame aprašyti visi ryšiai su paketais ir jų teikėjų failais, failas pildomas automatiškai, kai yra naudojama Composer programa, bet kadangi paketas buvo kūrimas lokaliai, paketo „serviceProvider.php“ failą reikėjo prijungti rankiniu būdu. Atlikus abu veiksmus, Laravel karkasas buvo pasirengęs dirbti su visais paketo failais.

Paketo failų struktūra buvo sukurta identiška Laravel karkaso struktūrai su MVC principu, tai reiškia, kad visi paketo failai buvo suskirstyti į 3 pagrindinius katalogus su konkrečiomis pareigomis. Modelių katalogas buvo atsakingas už klases, kurios atlieka veiksmus ir dirba su duomenų įrašymu ar skaitymu duomenų bazėje. Atvaizdavimo katalogas yra atsakingas už failus, kurie atlieka turinio vizualizavimą. Valdiklių katalogas, kuriame yra visos pagrindinės klasės su specifine

logika, duomenų apdorojimu, skaičiavimais ir CRUD generavimu. Papildomai, paketas turi atskirus katalogus maršrutams ir duomenų bazių migracijom saugoti. Sukurtas paketas turi keturias pagrindines funkcijas, kurios yra: migracijos generavimas, modelio generavimas, valdiklio generavimas ir vaizdo failo generavimas. Visos šios funkcijos turi atsakingas klases, kurios atlieka visą logiką ir grąžina reikiamą rezultatą, jos yra patalpintos Valdiklio kataloge.

Verta paminėti apie dvi programos kodo dizaino projektavimo strategijas, kurios buvo plačiai naudojamos kuriant paketą ir turėjo didelės įtakos paketo failų struktūrai. Pirmoji strategija buvo sukurti specialią klasę, kuri būtų atsakinga už įvairių tipų duomenų saugojimą ir tolesnį transportavimą. Tokio tipo klasės dažnai vadinamos duomenų perdavimo objektu (DTO). DTO yra naudingas, kai reikia sugrupuoti reikšmes ir perduoti duomenis skirtingiems procesams. Iš gryo dizaino perspektyvos DTO technika yra sprendimas, labai artimas tobulumui. Pakete DTO naudojamas duomenims su nustatymais perduoti tarp visų CRUD generatorių. Kita svarbi naudojama kodo projektavimo strategija yra gamyklinis modelis (angl. factory). Gamyklinis modelis yra atsakingas už visų naujų objektų inicijavimą su visomis priklausomybėmis, kurių reikalauja objektai. Dėl tokios technikos šaltinio kodas tampa patikimesnis, mažiau susietas ir lengvai išplečiamas. CRUD paketo kode pagrindinė valdiklio klasė turi ryšį su gamykliniu modeliu, kuris gali kurti naujus objektus ir grąžinti juos valdikliui tolimesniems procesams, išsamiau apie šaltinio kodą ir visas generatorių klases bus aprašoma tolimesniuose skyriuose.

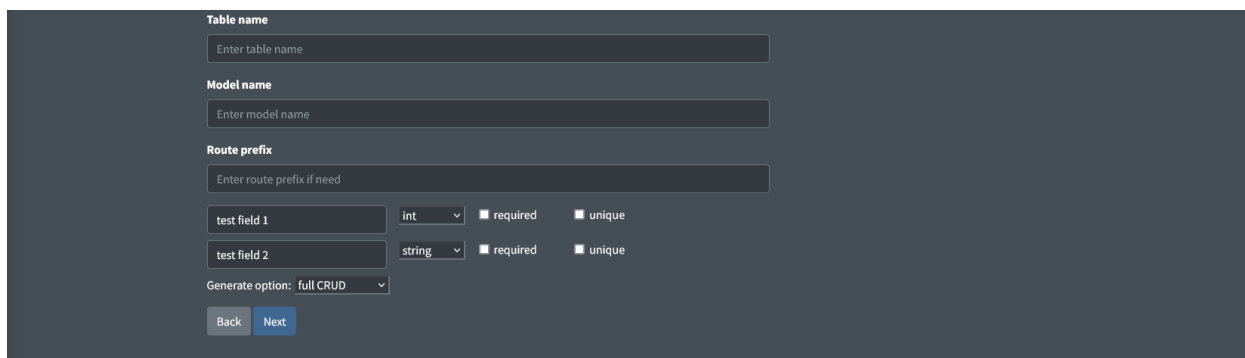
### 3.3. CRUD generavimo aprašymas

Viena iš pagrindinių paketo funkcijų yra CRUD generavimas. Kadangi paketas turi patogią grafinę sąsają, CRUD generavimas prasideda nuo formos pateikimo su nustatymais ir parametrais CRUD generavimui. Vartotojas turi įvesti lentelės pavadinimą, modelio pavadinimą ir pasirinkti kiekvieno lauko tipą. Galimi laukų tipai yra tokie: „date“, „int“, „string“, „text“. Formą galima pamatyti 8 pav. Formos duomenys visada tikrinami valdiklyje, jei atsiranda klaidų, naudotojas gauna pranešimą su klaidos tekstu. Valdiklis naudoja „DtoMapper“ klasę, kuri transformuoja formos duomenis į specialų „CrudParametersTransfer“ klasės objektą, kuris toliau yra naudojamas kiekvienoje generatorių klasėje. Duomenų perdavimo objekto pavyzdys vaizduojamas „C.1.“ priede. CRUD generavimo funkcija leidžia naudotojui generuoti atskirus CRUD elementus, todėl valdiklis patikrina, kurią generavimo parinktį pasirinko naudotojas ir, priklausomai nuo to, išskviečia atsakingo generatoriaus klasę. CRUD generavimo funkcija leidžia naudotojui generuoti atskirus CRUD elementus, todėl valdiklis patikrina, ką reikia sugeneruoti. Valdiklis ieško generavimo parinktį lauko, kuris taip pat pateikiamas su nustatymų formos duomenimis, priklausomai nuo naudotojo parinktį, valdiklis išskviečia atsakingo generatoriaus klasę, kuri sugeneruos pasirinktą CRUD elementą. Kai naudotojas pasirenka generuoti visą CRUD, valdiklis gauna generatoriaus klasių rinkinį, toliau kviečia kiekvieno generatoriaus metodą ir pateikia duomenų perdavimo objektą su nustatymais. CRUD valdiklį galima pamatyti „B.1.“ priede.

### 3.4. Migracijos generavimo aprašymas

Viena iš paketo funkcijų yra migracijos generavimas. Už šią funkciją yra atsakinga speciali generatoriaus klasė „MigrationGenerator“. Klasė turi du pagrindinius metodus generuoti migracijos failus. Skirtumas tarp metodų yra tas, kad „generate()“ metodas generuoja pradinį migracijos failą, kuris sukuria naują lentelę duomenų bazėje, o metodas „generateRelationshipMigration()“ generuoja migracijos failą su aprašytu sąryšiu. Sėkmingam darbui, reikia dviejų paruoštinių failų,

kurie proceso metu bus nukopijuoti, modifikuoti ir patalpinti kaip nauji migracijos failai. Paruoštinių failų turinys atvaizduojamas „A.2.“ ir „A.3.“ prieduose. Verta paminėti, kad „Migration Generator“ klasė generuoja tik migracijos failo turinį, toliau yra iškviečiama „FileWriter“ klasė, kuri sukuria reikiamą katalogą ir ten patalpina naują migracijos failą. Pagrindinis sunkumas kuriant šią klasę buvo rasti geriausią sąryšių aprašymo strategiją, kuri neribos naudotojo ir suteiks laisvę generuojant CRUD. Sprendimas buvo atskirti CRUD generavimo ir modelių sąryšių funkcijas. Dabartinė sukurto CRUD generatoriaus paketo versija palaiko tik keturių tipų kintamuosius, „date“, „integer“, „string“, „text“, kuriuos galima sugeneruoti. Tai yra pagrindinis migracijos generavimo funkcijos trūkumas, plačiau apie tai parašyta ateities darbų plane.

The image shows a web-based form for generating CRUD operations. It has a dark theme. The form is divided into several sections: 'Table name' with a text input field; 'Model name' with a text input field; 'Route prefix' with a text input field and a note 'Enter route prefix if need'; two test fields, 'test field 1' and 'test field 2', each with a type dropdown (currently set to 'int' and 'string' respectively) and checkboxes for 'required' and 'unique'; a 'Generate option' dropdown set to 'full CRUD'; and 'Back' and 'Next' buttons at the bottom.

8 pav. CRUD generavimo forma

### 3.5. Modelio generavimo aprašymas

Modelio generavimo funkcija yra labai panaši į anksčiau aprašytą migracijos generavimo funkciją. Už šią funkciją yra atsakinga „ModelGenerator“ klasė, kurios sėkmingam darbui reikalingas tik vienas paruoštinis failas. Paruoštinio failo turinys atvaizduojamas „A.4.“ priede. Klasė turi vieną pagrindinį metodą, kuris gauna naudotojo įvesties duomenis kaip duomenų perdavimo objektą ir generuoja modelio failo turinį. Kuriant failo turinį, klasė nukopijuoja paruoštinį failą ir pakeičia lentelės pavadinimą, modelio pavadinimą ir „fillable“ kintamąjį pagal naudotojo įvestį. Paskutiniame modelio generavimo etape, iškviečiama „FileWriter“ klasė, kuri sukuria failą tinkamame kataloge. Verta paminėti, kad sugeneruotas modelio failas neturi jokių metodų sąryšiams aprašyti, jie pridedami atskirai modelio sąryšių kūrimo procese. Šis procesas bus išsamiai aprašytas tolesniuose skyriuose.

### 3.6. Valdiklio generavimo aprašymas

Trečioji paketo funkcija yra valdiklio generavimas su kūrimo, naujinimo, skaitymo ir trynimo funkcijomis. Už šią funkciją yra atsakinga „ControllerGenerator“ klasė, nepaisant to, kad klasė turi panašią struktūrą kaip anksčiau aprašytos klasės, generavimo procese ji turi atlikti daugiau užduočių. Šiai klasei taip pat reikėjo sukurti paruoštinį failą su visomis kūrimo, skaitymo, atnaujinimo ir trynimo funkcijomis. Paruoštinio failo turinys atvaizduojamas „A.1.“ priede. „ControllerGenerator“ klasė pradžioje nukopijuoja visą paruoštinio failo turinį ir pakeičia pažymėtas vietas naudotojo įvestais duomenimis. Su tikslu įvesti daugiau supratimo, pateikiamas pavyzdys: paruoštiniame faile tose vietose, kur turėtų būti modelio pavadinimas, yra parašyta „{ {model name}}“, generatoriaus klasė suranda visas tokias vietas ir įrašo modelio pavadinimą iš naudotojo įvestų

duomenų. Pakeitus visas pažymėtas vietas, generatorius sukuria validacijos taisykles, kurios bus naudojamos sugeneruoto valdiklio skaitymo ir naujinimo funkcijose. Validacijos taisyklės kuriamos atsižvelgiant į kelis aspektus. Pirmiausia generatorius patikrina pasirinktas kiekvieno lauko validacijos taisykles iš naudotojo įvesties duomenų. Dabartinė CRUD generatoriaus versija leidžia naudotojui pridėti tik privalomo ir unikalaus lauko validacijos taisykles. Papildomai, generatoriaus klasė pridės numatytąsias kiekvieno lauko tipo validacijos taisykles. Patikslinant, pateikiamas pavyzdys: „string“ tipo laukui bus pridėta teksto validacijos taisyklė, „int“ tipo atitinkami bus pridėta skaičiaus validacijos taisyklė. Kaip ir anksčiau aprašytose generatorių klasėse, „ControllerGenerator“ naudoja „FileWriter“ klasę, kuri sukuria valdiklio failą. Verta paminėti, kad „FileWriter“ klasė taip pat naudojama kuriant naujus valdiklio maršrutus. Tai paskutinis valdiklio generavimo etapas, „FileWriter“ klasė suranda numatytąjį maršrutų failą „web.php“ ir įrašo naują maršrutą į esamų maršrutų sąrašą. Pagrindinis sunkumas kuriant „ControllerGenerator“ klasę buvo nuspręsti, kaip sukurti universalią validacijos taisyklių pridėjimo sistemą, kuri galės taikyti visas įmanomas validacijos taisykles. Dabartinis sprendimas yra labai ribotas, dėl to bus reikalingas pertvarkymas. Plačiau apie tai parašyta ateities darbų plane.

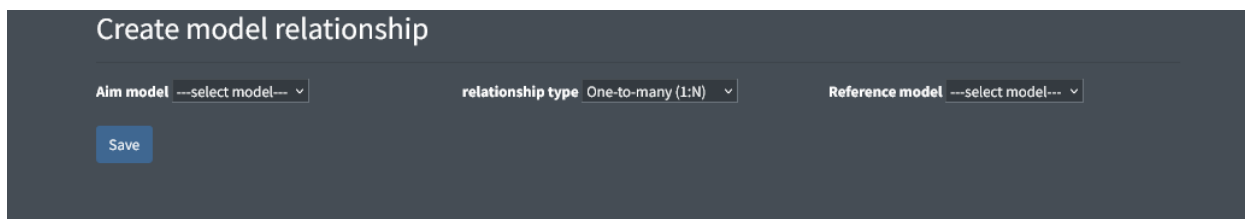
### 3.7. Vaizdų generavimo aprašymas

Norint visiškai automatizuoti CRUD generavimą, taip pat reikėjo sukurti vaizdo failus su duomenų kūrimo ir modifikavimo formomis. Šiai funkcijai buvo sukurta atsakinga generatoriaus klasė, vadinama „ViewGenerator“. Klasė generuoja keturis vaizdo failus, skirtus CRUD funkcijoms – generuojamų duomenų atvaizdavimui, kūrimui, atnaujinimui ir trynimui. Sėkmingam generatoriaus klasės darbui reikalingi trys paruoštiniai failai – „form-skeleton“, „index-skeleton“, „show-skeleton“. Kiekvieno paruoštinio failo turinys vaizduojamas „A.5.“, „A.6.“, „A.7.“ prieduose. Panašiai kaip ir anksčiau aprašytos generatorių klasės, „ViewGenerator“ nukopijuoja paruoštinio failo turinį, jį modifikuoja ir naudodamas „FileWriter“ klasę sukuria naują vaizdo failą. Generatorius naudoja „index-skeleton“ failą tam, kad sukurtų pagrindinį vaizdo failą, kuriame bus rodomas pilnas sukurtų įrašų sąrašas. Sugeneruotas vaizdo failas bus įėjimo taškas į visas sugeneruoto modelio CRUD funkcijas. Kuriant CRUD labai dažnai yra naudojamos du vaizdo failai su atskiromis formomis duomenims kurti ir atnaujinti, tokia strategija visada atneša daug pasikartojančių kodo eilučių. Siekiant išvengti tokios situacijos, buvo nuspręsta sukurti vieną vaizdo failą su forma, kuri priklausomai nuo nurodytų parametrų galės dirbti tiek su duomenų kūrimu, tiek atnaujinimu. Tokiam vaizdo failo kūrimui generatoriaus klasė naudoja „form-skeleton“ paruoštinį failą. Paskutinis paruoštinis failas „show-skeleton“ naudojamas generuoti vaizdo failą, kuriame bus rodoma visa išsami informacija apie pasirinktą įrašą. Pagrindinis „ViewGenerator“ klasės trūkumas yra tas, kad dabartinė versija palaiko tik formas su „input“, „textarea“ id „date“ įvesties tipais. Plačiau apie tai parašyta ateities darbų plane.

### 3.8. Modelių sąryšių valdiklio aprašymas

CRUD generatoriaus paketas turi funkciją, leidžiančią kurti sąryšius tarp sugeneruotų objektų ir valdyti susijusius objektus. Visos funkcijos, kurios yra susijusios su modelio sąryšiu valdymu, yra patalpintos į atskirą valdiklio klasę „ModelRelationshipController“. Valdiklio klasės turinys pateikiamas „B.2.“ priede. naudotojas kurdamas naują sąryšį tarp sugeneruotų modelių turi pateikti formą 9 pav. su modelių pavadinimais ir ryšio tipu. Pateiktos formos duomenys tikrinami valdiklio klasėje, jei nėra klaidų, valdiklis iškviečia „ModelRelationshipManager“ klasę, kuri faktiškai

sukuria ryšį. Naujo sąryšio kūrimas tarp sugeneruotų modelių atliekamas dviem etapais. Pirmasis žingsnis yra sukurti migraciją, kuri pridės naują sąryšio lauką į susijusio modelio lentelę. Antrasis žingsnis yra įtraukti reikalingus sąryšio metodus prie susijusių modelių klasių. Patikslinant, pateikiamas pavyzdys: sąryšio vienas su daugeliu atveju pagrindinė modelio klasė bus papildyta „hasMany()“ metodu, susijusi modelio klasė bus papildyta „belongsTo()“ metodu. Sėkmingai sukūrus naują sąryšį, jis bus rodomas sąryšių sąrašė, papildomai, naudotojas galės pereiti į sąryšių valdymo puslapį, kuriame galės priskirti arba panaikinti susijusių modelių objektus. Kaip pagrindinį sąryšių kūrimo funkcijos trūkumą paminėčiau tai, kad dabartinė paketo versija leidžia kurti tik vieno su daugeliu tipo sąryšius, plačiau apie tai parašyta ateities darbų plane.



9 pav. Naujo sąryšio kūrimo forma

### 3.9. Konfigūruojamo puslapiavimo aprašymas

CRUD generatoriaus paketas palaiko konfigūruojamo puslapiavimo funkciją. Naudotojas gali pasirinkti puslapiavimo nustatymus trijose skirtingose vietose – sąryšių sąrašė, sugeneruotų CRUD sąrašė, CRUD elementų sąrašė. Naudotojui teisingai pateikus formą su nustatymais, jie yra saugomi „config.json“ faile, kuris sukuriamas paketo diegimo metu. Puslapiavimo nustatymų duomenų skaitymui buvo sukurta atskira klasė „CrudConfig“, kuri turi tokius metodus: „getDefaultPagination()“ - gauti numatytąjį elementų skaičių puslapyje, „getCrudListPagination()“ - gauti elementų skaičių sugeneruotų CRUD sąrašo puslapyje, „getRelationshipListPaginaton()“ - gauti elementų skaičių sąryšių puslapyje, „getCrudElementPagination()“ - gauti elementų skaičių pasirinkto CRUD sąrašo puslapyje. Pilnas „CrudConfig“ klasės turinys parodytas „C.2.“ priede. Puslapiavimo funkcijos atskyrimas leido gauti švarų, daugkartinį ir nesikartojantį kodą valdikliuose, kas žymiai pagerino paketo veikimą ir programinio kodo skaitomumą.

### 3.10. Paketo publikavimas

Paketo publikavimas yra procesas, kai visas paketo šaltinio kodas įkeliamas į Packagist platformą. Šis procesas taip pat gali būti vadinamas viešas paketo paskelbimas, nes po sėkmingo paketo paskelbimo jis tampa visiems viešai prieinamas per *composer require* komandą. Pirmasis paketo paskelbimo proceso žingsnis buvo sukurti naują Github repositoriją, į kurią buvo perkeltas visas paketo šaltinio kodas. Sėkmingam publikavimui svarbu, kad repositoriją būtų nustatyta kaip vieša, kitaip Packagist platforma paketo nesuras. Papildomas žingsnis buvo sukurti naują leidimą su paketo versijos numeriu ir aprašymu. Pilnai paruoštą Github repositoriją galima peržiūrėti adresu - <https://github.com/RafalKLab/crud>. Paskutinis žingsnis buvo prisijungti prie Packagist ir pateikti Github repositorijos nuorodą. Patvirtinus paketo pavadinimą, paketas tapo visiems viešai prieinamas per *composer require rklab/crud* komandą.

## Išvados

Darbo metu buvo atlikta viešai prieinamų Laravel karkaso CRUD generatorių paketų analizė, išskiriant kiekvieno privalumus bei trūkumus. Remiantis analizės rezultatais, buvo suformuluoti pagrindiniai kriterijai, poreikiai, funkciniai ir nefunkciniai reikalavimai CRUD generavimo paketui, bei iškeltas darbo tikslas. Darbo tikslas buvo sukurti viešai prieinamą CRUD funkcijų generavimo paketą Laravel karkasui, kuris leistų automatiškai generuoti migracijas, modelius, valdiklius ir vaizdo failus. Suprojektavus teorinį sistemos modelį ir sėkmingai sukūrus programinę dalį, darbo tikslas buvo pasiektas. Atsižvelgiant į atlikto darbo rezultatus buvo sudarytos tokios išvados:

- CRUD generatorių analizės metu pastebėta, kad generatoriai su didžiuliu funkcijų sąrašu dažnai turi sudėtingą sąsają, kuri gali daryti neigiamą įtaką naudotojui. Tai reiškia, kad labai svarbu neperkrauti generatoriaus nenaudingomis funkcijomis, kurios veikia tik kraštutinėse situacijose.
- Atlikus CRUD generatorių analizę buvo pastebėta, kad paketui yra labai svarbu turėti greitą ir nesudėtingą diegimo procesą. Tai reiškia, kad reikia stengtis kurti paketą kaip nepriklausomą įrankį ir vengti didžiulio priklausomų paketų kiekio, kitaip tai gali pailginti diegimo procesą.
- Programuojant sistemą buvo pastebėta, jog yra itin svarbų rašyti švarų kodą, laikantis SOLID principų ir visų geriausių praktikų bei PSR rekomendacijų. Dėl to programos kodas bus lengviau skaitomas, naudojamas pakartotinai, o sistemos priežiūros procesas bus lengvesnis.
- Analizuojant Laravel paketų kūrimo dokumentaciją buvo pastebėta, kad yra labai svarbu naudoti „publishes()“ metodą paketo failams, kuriuos naudotojas gali norėti išplėsti. Tokie failai diegimo proceso metu bus perkelti į naudotojo projekto darbo katalogą, o tai suteiks galimybę naudotojui laisvai keisti grafinės sąsajos stilius ar pridėti naują funkcionalumą.
- Analizuojant Packagist paketų įkėlimo dokumentaciją buvo pastebėta, kad labai svarbu tinkamai valdyti paketo versijų pavadinimus, kurie turi atitikti semantinę versijų pavadinimų kūrimo strategiją. Tai reiškia, kad paketo versijos pavadinimas turi išlaikyti šabloną „X.Y.Z“, kur X reiškia pagrindinę versiją, kai atliekamas nesuderinamas pakeitimas su ankstesne paketo versija, Y reiškia nedidelę versiją, kai pridamas funkcionalumas išlaikant suderinamumą su ankstesne paketo versija, Z reiškia klaidų taisymo versiją, kai atliekamas programinio kodo klaidų taisymas išlaikant suderinamumą su ankstesne paketo versija.

## Ateities darbų planas

Siekiant išplėsti CRUD generatoriaus funkcionalumą ir padidinti potencialių naudotojų skaičių, pateikiamas ateities darbų planas. Paketui siūlomi tokie ateities darbai:

- Išplėsti CRUD generatoriaus paketo migracijos failo generavimo funkciją, kad būtų palaidomi tokie laukų tipai kaip: „double“, „boolean“, „enum“, „char“.
- Išplėsti CRUD generatoriaus paketo sąryšių kūrimo funkcionalumą, kad būtų galima kurti vienas su vienu (1:1) ir daugelis su daugeliu (N:N) tipo sąryšius.
- Sukurti CRUD generatoriaus vaizdo failų formų kūrimo mechanizmą, kuris, atsižvelgiant į kintamųjų tipus, formoje sudarys atitinkamus įvesties tipus. Aiškumo dėlei pateikiamas pavyzdys: „boolean“ tipo kintamajam bus priskirtas įvesties tipas „checkbox“, o „enum“ tipo kintamajam bus priskirtas įvesties tipas „select“.
- Sukurti validacijos taisyklių pridėjimo sistemą, kuri leis naudotojui pridėti tokias validacijos taisykles kaip: mažiausia ir didžiausia kintamojo reikšmė, teksto ilgis, elektroninis paštas, telefono numeris.

## Literatūros šaltiniai

- [1] crestapps/laravel-code-generator. <https://crestapps.com/laravel-code-generator/docs/2.2> [lankytasi 17/12/2022].
- [2] infyomlabs/laravel-generator. <https://packagist.org/packages/infyomlabs/laravel-generator> [lankytasi 01/12/2022].
- [3] Larafast/fastapi. <https://dev.to/mahmouditaly/larafast-fastapi-package-egh> [lankytasi 22/12/2022].
- [4] Laravel paketų kūrimo dokumentacija. <https://laravel.com/docs/9.x/packages> [lankytasi 23/12/2022].
- [5] Packagist paketų įkėlimo dokumentacija. <https://packagist.org/about> [lankytasi 29/12/2022].
- [6] roland/crud. <https://packagist.org/packages/roland/crud> [lankytasi 06/12/2022].
- [7] Robert C. Martin. Clean code: A handbook of agile software craftsmanship, 2008. [https://thixalongmy.haugiang.gov.vn/media/1175/clean\\_code.pdf](https://thixalongmy.haugiang.gov.vn/media/1175/clean_code.pdf) [lankytasi 02/12/2022].

# Priedai

## A. Paketo paruoštiniai failai

### A.1. Valdiklio paruoštinis failas

```
<?php

namespace App\Http\Controllers\{{model name}};

use App\Http\Controllers\Controller;
use App\Models\{{model name}}\{{model name}};
use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Rklab\Crud\Http\Controllers\Config\CrudConfig;

class {{model name}}Controller extends Controller
{
    /**
     * @return View
     */
    public function index(): View
    {
        $itemsPerPage = $this->getItemsPerPage();

        ${{model name lowercase}}s = {{model name}}::paginate($itemsPerPage);

        return view('{{model name lowercase}}.index')
            ->with('{{model name lowercase}}s', ${{model name lowercase}}s);
    }

    /**
     * @return View
     */
    public function create(): View
    {
        return view('{{model name lowercase}}.form');
    }

    /**
     * @param Request $request
     *
     * @return RedirectResponse
     */
    public function store(Request $request): RedirectResponse
    {
        {{validation}}
```

```

        {{model name}}::create($request->all());

        return redirect()->route('{{model name lowercase}}s.index')
        ->with('success','Successfully added a new {{model name}}.');
    }

    /**
     * @param {{model name}} ${{model name lowercase}}
     *
     * @return View
     */
    public function show('{{model name}} ${{model name lowercase}}): View
    {
        return view('{{model name lowercase}}.show',
            compact('{{model name lowercase}}'));
    }

    /**
     * @param {{model name}} ${{model name lowercase}}
     *
     * @return View
     */
    public function edit('{{model name}} ${{model name lowercase}}): View
    {
        return view('{{model name lowercase}}.form',
            compact('{{model name lowercase}}'));
    }

    /**
     * @param {{model name}} ${{model name lowercase}}
     *
     * @return RedirectResponse
     */
    public function update(
        Request $request,
        {{model name}} ${{model name lowercase}}
    ): RedirectResponse {
        {{validation}}

        ${{model name lowercase}}->update($request->all());

        return redirect()->route('{{model name lowercase}}s.index');
    }

    /**
     * @param {{model name}} ${{model name lowercase}}
     *
     * @return RedirectResponse

```

```

*/
public function destroy(
    {{model name}} ${{model name lowercase}}
): RedirectResponse {
    ${{model name lowercase}}->delete();

    return redirect()->route('{{model name lowercase}}s.index');
}

/**
 * @return int
 */
private function getItemsPerPage(): int
{
    return (new CrudConfig())
        ->getCrudElementPagination();
}
}

```

## A.2. Migracijos paruoštinis failas

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        {{% migration-up-method %}}
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        {{% migration-down-method %}}
    }
}

```

```
};
```

### A.3. Sąryšių migracijos paruoštinis failas

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('{{table name}}', function (Blueprint $table) {
            {{relation field}}
        });
    }
};
```

### A.4. Modelio paruoštinis failas

```
<?php

namespace App\Models\{{modelName}};

use Illuminate\Database\Eloquent\Model;

class {{modelName}} extends Model
{
    /**
     * The database table used by the model.
     *
     * @var string
     */
    protected $table = '{{tableName}}';

    /**
     * Attributes that should be mass-assignable.
     *
     * @var array
     */
}
```

```
    */  
    protected $fillable = {{fillable}};  
  
    // This is auto generated tag do not delete it  
    // model relationship  
  
}
```

## A.5. Pagrindinio vaizdo failo paruoštinis failas

```
@extends('crud::template.default')
@section('content')
    <div class="container">

        <h1>{{model name lowercase}}s</h1>
        <table class="table">
            <thead>
                <tr>
                    <th scope="col">#</th>
                    {{table fields}}
                    <th scope="col">action</th>
                </tr>
            </thead>
            <tbody>
                @foreach(${{model name lowercase}}s as ${{model name lowercase}})
                    <tr>
                        <td>{{${{model name lowercase}}->id}}</td>
                        {{table fields data}}
                        <td>
                            <form
                                action="{{route('{{model name lowercase}}s.destroy',
                                    ${{model name lowercase}})}}"
                                method="POST">
                                <a href="{{route('{{model name lowercase}}s.show',
                                    ${{model name lowercase}})}}"
                                    class="btn btn-light">
                                    show
                                </a>
                                <a href="{{route('{{model name lowercase}}s.edit',
                                    ${{model name lowercase}})}}" class="btn btn-info">
                                    edit
                                </a>
                                @csrf
                                @method('DELETE')
                                <input type="submit"
                                    class="btn btn-danger"
                                    value="Delete">
                                </form>
                            </td>
                        </tr>
                    @endforeach
                </tbody>
            </table>
            {{ ${{model name lowercase}}s->links() }}
            <a href="{{route('{{model name lowercase}}s.create')}}"
                class="btn btn-success">Add new</a>
        </div>
```

```
@endsection
```

## A.6. Formos paruoštinis failas

```
@extends('crud::template.default')

@section('content')
    <div class="container">
        @isset(${{model name lowercase}})
            <h1>{{model name lowercase}} update</h1>
        @else
            <h1>Add new {{model name lowercase}}</h1>
        @endisset
        <hr>
        <form method="POST"
            @isset(${{model name lowercase}})
                action="{{route('{{model name lowercase}}s.update',
                    {{model name lowercase}})}}"
            @else
                action="{{route('{{model name lowercase}}s.store')}}"
            @endisset
        >
            <div>
                @isset(${{model name lowercase}})
                    @method('PUT')
                @endisset
                @csrf

                {{form inputs}}
                <a class = "btn btn-secondary"
                    href="{{route('{{model name lowercase}}s.index')}}">
                    Back
                </a>
                <button class="btn btn-success">Save</button>
            </div>
        </form>
    </div>
@endsection
```

## A.7. Informacijos vaizdo failo paruoštinis failas

```
@extends('crud::template.default')
@section('content')
    <div class="container">

        <table class="table table-striped">
```

```

        <thead>
        <tr>
            <th scope="col">#</th>
            {{model fields}}
        </tr>
        </thead>
        <tbody>
        <tr>
            <td>{{${{model name lowercase}}->id}}</td>
            {{model fields data}}
        </tr>
        </tbody>
    </table>

    <a class = "btn btn-secondary"
    href="{{route('{{model name lowercase}}s.index')}}">
    Back
    </a>
</div>

```

@endsection

## B. Sistemų valdikliai

### B.1. CrudController valdiklis

```

<?php

namespace Rklab\Crud\Http\Controllers;

use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Artisan;
use Illuminate\Support\Facades\Session;
use Rklab\Crud\Dto\CrudParametersTransfer;
use Rklab\Crud\Http\Controllers\Generator\CrudGeneratorInterface;
use Rklab\Crud\Models\Crud;

class CrudController extends Controller
{
    /**
     * @return View
     */
    public function prepareFields(): View
    {

```

```

        return view('crud::crud.crud_fields');
    }

    /**
     * @param Request $request
     *
     * @return RedirectResponse
     */
    public function generateFields(Request $request): RedirectResponse
    {
        $this->validate($request, [
            'number_of_fields' => 'numeric|min:1|max:20',
        ]);
        Session::put('number_of_fields', $request->input('number_of_fields'));

        return redirect()->route('generate');
    }

    /**
     * @return View
     */
    public function prepareCrud(): View
    {
        return view("crud::crud.crud_settings");
    }

    /**
     * @param Request $request
     *
     * @return RedirectResponse
     */
    public function generateCrud(Request $request): RedirectResponse
    {
        for ($i = 1; $i <= $request->input('fieldItterator'); $i++) {
            $fieldName = 'field_name_' . $i;
            $selectType = 'select_type_' . $i;
            $this->validate($request, [
                $fieldName => 'required|alpha_dash|max:100',
                $selectType => 'required|alpha_dash|max:100',
                'table_name' => 'required|alpha|max:25|unique:cruds',
                'model_name' => 'required|alpha|max:25|unique:cruds',
                'route_prefix' => 'nullable|alpha|max:100|not_in:crud',
            ]);
        }

        $transfer = $this->getTransfer($request);

        match ($transfer->getGenerateOption()) {
            'full' => $this->generateFullCrud($transfer),
        }
    }

```

```

        'migration' => $this->generateOnlyMigration($transfer),
        'model' => $this->generateOnlyModel($transfer),
        'controller' => $this->generateOnlyController($transfer),
        'view' => $this->generateOnlyView($transfer),
    };

    return redirect()->route('dashboard')
        ->with('success','Successfully generated new CRUD.');
```

}

/\*\*
 \* @param CrudParametersTransfer \$transfer
 \*/
private function saveCrud(CrudParametersTransfer \$transfer): void
{
 \$params = [
 'route' => sprintf(
 "%ss.index",
 strtolower(\$transfer->getModelName())
 ),
 'table\_name' => \$transfer->getTableName(),
 'model\_name' => \$transfer->getModelName(),
 ];

 Crud::create(\$params);
}

/\*\*
 \* @return View
 \*/
public function listCrud(): View
{
 \$cruds = Crud::paginate(10);

 return view('crud::crud.crud\_list')->with('cruds', \$cruds);
}

/\*\*
 \* @param Request \$request
 \*
 \* @return CrudParametersTransfer
 \*/
private function getTransfer(Request \$request): CrudParametersTransfer
{
 return \$this->getCrudFactory()->createDtoMapper()
 ->mapMigrationParametersToDto(
 \$this->getCrudFactory()
 ->createMigrationTableParametersTransfer(),
 \$request,
 );
}

```

        );
    }

    /**
     * @return CrudGeneratorInterface[]
     */
    private function getCrudGenerators(): array
    {
        return $this->getCrudFactory()
            ->getGeneratorCollection()
            ->getGenerators();
    }

    /**
     * @param CrudParametersTransfer $transfer
     *
     * @return CrudParametersTransfer
     */
    private function generateOnlyMigration(CrudParametersTransfer $transfer)
    : CrudParametersTransfer {
        $this->getCrudFactory()
            ->createMigrationGenerator()
            ->generate($transfer);

        return $transfer;
    }

    /**
     * @param CrudParametersTransfer $transfer
     *
     * @return CrudParametersTransfer
     */
    private function generateOnlyModel(CrudParametersTransfer $transfer)
    : CrudParametersTransfer {
        $this->getCrudFactory()
            ->createModelGenerator()
            ->generate($transfer);

        return $transfer;
    }

    /**
     * @param CrudParametersTransfer $transfer
     *
     * @return CrudParametersTransfer
     */
    private function generateOnlyController(CrudParametersTransfer $transfer)
    : CrudParametersTransfer {
        $this->getCrudFactory()

```

```

        ->createControllerGenerator()
        ->generate($transfer);

        return $transfer;
    }

    /**
     * @param CrudParametersTransfer $transfer
     *
     * @return CrudParametersTransfer
     */
    private function generateOnlyView(CrudParametersTransfer $transfer)
    : CrudParametersTransfer {
        $this->getCrudFactory()
            ->createViewGenerator()
            ->generate($transfer);

        return $transfer;
    }

    /**
     * @param CrudParametersTransfer $transfer
     *
     * @return CrudParametersTransfer
     */
    private function generateFullCrud(CrudParametersTransfer $transfer)
    : CrudParametersTransfer {
        foreach ($this->getCrudGenerators() as $generator) {
            $generator->generate($transfer);
        }

        $this->saveCrud($transfer);

        Artisan::call('migrate');

        return $transfer;
    }
}

```

## B.2. ModelRelationshipController valdiklis

```

<?php

namespace Rklab\Crud\Http\Controllers;

use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;

```

```

use Illuminate\Support\Facades\Artisan;
use Illuminate\Support\Facades\Redirect;
use Rklab\Crud\dto\ModelRelationshipTransfer;
use Rklab\Crud\Models\RelatedModel;

class ModelRelationshipController extends Controller
{
    /**
     * @return View
     */
    public function index(): View
    {
        $relatedModels = $this->getRepository()
            ->getRelatedModels();

        return view("crud::relationship.index")
            ->with('relatedModels', $relatedModels);
    }

    /**
     * @param RelatedModel $model
     *
     * @return \Illuminate\Contracts\Foundation\Application
     *         |\Illuminate\Contracts\View\Factory|View|RedirectResponse
     */
    public function show(RelatedModel $model)
    : View | RedirectResponse {
        $fullyQualifiedAimModelClassName = $this->getFullyQualifiedClassName(
            $model->aim_model
        );

        $modelData = $fullyQualifiedAimModelClassName::all();
        if ($modelData) {
            $modelDataFieldNames = $modelData[0]->getFillable();

            return view("crud::relationship.show")
                ->with(compact('modelData', 'modelDataFieldNames', 'model'));
        } else {
            return redirect()
                ->route('dashboard')
                ->with('warning', 'Model data not found!');
        }
    }

    /**
     * @param Request $request
     *
     * @return View|RedirectResponse
     */

```

```

*/
public function getAssignTable(Request $request)
: View | RedirectResponse {
    $aimModelId = $request->id;
    $aimModel = $request->aim_model_name;
    $refModel = $request->ref_model_name;

    $aimModelField = sprintf("%s_id", strtolower($aimModel));
    $fullyQualifiedRefClassName = $this->getFullyQualifiedClassName(
        $refModel
    );

    $assignedData = $fullyQualifiedRefClassName::where(
        $aimModelField, $aimModelId
    )->get();

    $unAssignedData = $fullyQualifiedRefClassName::where(
        $aimModelField, null)
    )->get();

    $refModelRecords = $fullyQualifiedRefClassName::skip(0)
    )->take(1)
    )->get();

    if ($refModelRecords) {
        $refModelFieldNames = $refModelRecords[0]
        )->getFillable();

        return view("crud::relationship.assign")->with(
            compact(
                'assignedData',
                'unAssignedData',
                'refModelFieldNames',
                'aimModelId',
                'aimModel',
                'refModel',
            )
        );
    } else {
        return redirect()
        )->route('dashboard')
        )->with('warning','Model data not found!');
    }
}

/**
 * @param Request $request
 *
 * @return RedirectResponse

```

```

*/
public function assign(Request $request)
: RedirectResponse {
    $fullyQualifiedRefClassName = $this->getFullyQualifiedClassName(
        $request->ref_model_name
    );
    $aimModelField = sprintf(
        "%s_id",
        strtolower($request->aim_model_name)
    );
    $refModelId = $request->ref_id;
    $aimModelId = $request->aim_id;

    $fullyQualifiedRefClassName::where('id', $refModelId)
    ->update([$aimModelField => $aimModelId]);

    return Redirect::back()
    ->with('success', "Model object was assigned.");
}

/**
 * @param Request $request
 *
 * @return RedirectResponse
 */
public function unAssign(Request $request): RedirectResponse
{
    $fullyQualifiedRefClassName = $this
    ->getFullyQualifiedClassName(
        $request->ref_model_name
    );

    $aimModelField = sprintf(
        "%s_id", strtolower($request->aim_model_name)
    );
    $refModelId = $request->ref_id;
    $aimModelId = $request->aim_id;

    $fullyQualifiedRefClassName::where(
        'id', $refModelId)
    ->where($aimModelField, $aimModelId)
    ->update([$aimModelField => null]);

    return Redirect::back()
    ->with('success', "Model object was unassigned.");
}

/**
 * @return View

```

```

    */
    public function createRelationship(): View
    {
        $cruds = $this->getRepository()->getCruds();

        return view("crud::relationship.form")->with('cruds', $cruds);
    }

    /**
     * @param Request $request
     *
     * @return RedirectResponse
     */
    public function storeRelationship(Request $request)
    : RedirectResponse {
        $this->validate($request, [
            'aim_model' => 'required',
            'relationship_type' => 'required',
            'ref_model' => 'required|different:aim_model',
        ]);

        $transfer = $this->getRelationshipTransfer($request);

        match ($request->input('relationship_type')) {
            '1:N' => $this->createOneToManyRelationship($transfer),
        };

        Artisan::call('migrate');

        $this->saveRelationEntry($transfer);

        return redirect()
            ->route('dashboard')
            ->with('success', 'New relation was created!');
    }

    /**
     * @param ModelRelationshipTransfer $transfer
     *
     * @return ModelRelationshipTransfer
     */
    private function createOneToManyRelationship(
        ModelRelationshipTransfer $transfer
    )
    : ModelRelationshipTransfer{
        $this->getCrudFactory()
            ->createOneToManyRelationshipManager()
            ->createRealation($transfer);
    }

```

```

        return $transfer;
    }

    /**
     * @param Request $request
     *
     * @return ModelRelationshipTransfer
     */
    private function getRelationshipTransfer(Request $request)
    : ModelRelationshipTransfer{
        return $this->getCrudFactory()->createDtoMapper()
            ->mapModelRelationshipToDto(
                $this->getCrudFactory()->createModelRelationshipTransfer(),
                $request,
            );
    }

    /**
     * @param ModelRelationshipTransfer $transfer
     */
    private function saveRelationEntry(ModelRelationshipTransfer $transfer
    ): void {
        $params = [
            'aim_model' => $transfer->getAimModelName(),
            'ref_model' => $transfer->getRefModelName(),
            'relation_type' => $transfer->getRelationType(),
        ];

        RelatedModel::create($params);
    }

    /**
     * @param string $modelName
     *
     * @return string
     */
    private function getFullyQualifiedClassName(string $modelName): string
    {
        return sprintf("App\Models\%1$s\%1$s", $modelName);
    }
}

```

## C. Paketo programinio kodo pavyzdžiai

### C.1. Duomenų perdavimo objektas (DTO)

```
Rklab\Crud\dto\CrudParametersTransfer [#1408 ▼ // packages\rklab\crud\src\Http\Controllers\Generator\Migration\MigrationGenerator.php:26
-tableName: "TestTableName"
-modelName: "TestModelName"
-generateOption: "full"
-routePrefix: "TestPrefix"
-tableFields: array:2 [▼
  0 => Rklab\Cr\FieldTransfer [#1408 ▼
    -fieldName: "test_field_1"
    -fieldType: "int"
    -fieldValidations: array:2 [▼
      0 => "required"
      1 => "unique"
    ]
  ]
  1 => Rklab\Cr\FieldTransfer [#1403 ▼
    -fieldName: "test_field_2"
    -fieldType: "string"
    -fieldValidations: array:2 [▼
      0 => "required"
      1 => "unique"
    ]
  ]
]
-validationRules: []
}
```

10 pav. Duomenų perdavimo objektas su įvesties duomenų pavyzdžiu

### C.2. CrudConfig.php failas

```
<?php
```

```
namespace Rklab\Crud\Http\Controllers\Config;
```

```
class CrudConfig
```

```
{
```

```
    public const ROUTE_PREFIX = 'generated';
```

```
    public const PAGINATION = 'pagination';
```

```
    public const CRUD_LIST = 'crud_list_pagination';
```

```
    public const CRUD_ELEMENTS_LIST = 'crud_elements_pagination';
```

```
    public const RELATIONSHIP_LIST = 'relationships_list_pagination';
```

```
    protected const DEFAULT_PAGINATION = 10;
```

```
    protected array $pagination;
```

```
    /**
```

```
     * CrudConfig constructor.
```

```
     */
```

```
    public function __construct()
```

```
    {
```

```
        $this->pagination = json_decode(
```

```
            file_get_contents(__DIR__.'/config.json'),
```

```
            true
```

```
        );
```

```
    }
```

```
    /**
```

```
     * @return string
```

```

        */
public function getDefaultRoutePrefix(): string
{
    return self::ROUTE_PREFIX;
}

/**
 * @return int
 */
public function getDefaultPagination(): int
{
    return self::DEFAULT_PAGINATION;
}

/**
 * @return int
 */
public function getCrudListPagination(): int
{
    return $this->pagination[self::PAGINATION][self::CRUD_LIST]
        ? $this->pagination[self::PAGINATION][self::CRUD_LIST]
        : $this->getDefaultPagination();
}

/**
 * @return int
 */
public function getRelationshipListPaginaton(): int
{
    return $this->pagination[self::PAGINATION][self::RELATIONSHIP_LIST]
        ? $this->pagination[self::PAGINATION][self::RELATIONSHIP_LIST]
        : $this->getDefaultPagination();
}

/**
 * @return int
 */
public function getCrudElementPagination(): int
{
    return $this->pagination[self::PAGINATION][self::CRUD_ELEMENTS_LIST]
        ? $this->pagination[self::PAGINATION][self::CRUD_ELEMENTS_LIST]
        : $this->getDefaultPagination();
}
}

```