KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Algirdas Simutis, Viktorija Petrošiūtė

KOMPONENTINIŲ INTERNETO SISTEMŲ KŪRIMO KARKASAS

Magistro darbas

Vadovas
doc. dr. L. Nemuraitė

KAUNAS, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

KOMPONENTINIŲ INTERNETO SISTEMŲ KŪRIMO KARKASAS

Magistro darbas

Vadovas
doc. dr. L. Nemuraitė

Recenzentas
doc. dr. E. Karčiauskas

Atliko
IFM 0/4 gr. studentai
A. Simutis,
V. Petrošiūtė

KAUNAS, 2006

THE FRAMEWORK FOR COMPONENT-BASED INTERNET SYSTEM CREATION USING PHP5

The creation of systems for Internet use is usually susceptible to problems such as complicated implementation and inefficient exploitation of money and time. The main problem is a discordant needs of customers and system developers: customers want to have a system, which works properly at the lowest price, while system developers want to maintain a high quality in development and to put the product on the market as soon as possible. To solve these problems we offer a framework, which is created according to principles of component based software engeneering combined with pattern oriented software architecture. A framework can be used as a foundation for extensible, flexible, easy to maintain and user-friendly systems for Internet use.

TURINYS

Lentelių sąrašas

Paveikslėlių sąrašas

ĮVADAS

Tobulėjant informacinėms sistemoms ir pasaulinio tinklo teikiamoms galimybėms, beveik būtinybe tapo turėti savo interneto svetainę ar sistemą. Šiandien pasaulyje yra sukurta šiek tiek daugiau nei 3 milijardai tinklalapių. Nuosavas informacinis portalas ne tik madinga, bet ir patogu, todėl vis daugiau įmonių, organizacijų ar pavienių asmenų pateikia informaciją apie save ar savo siūlomą produkciją internetinių sistemų pagalba. Tačiau tenka pripažinti, kad sudėtingesnių internetinių sistemų kūrimas – netriviali paslauga, kuri reikalauja ne tik daug laiko bei pinigų, bet ir gilaus supratimo apie kuriamą sistemą.

Kalbant apie kuriamos sistemos suvokimą, žinome, kad tam reikia išanalizuoti sistemos poreikius bei sukūrimo ir realizavimo galimybes. Šiam tikslui įgyvendinti, neišvengiamai reikalinga aukštą kvalifikaciją turinti darbo jėga, todėl šios aplinkybės, kuri yra būtina norint sukurti tinkamą informacinę sistemą, atsisakyti negalima.. Analizuodami patį kūrimo procesą matome, kad čia laikas ir pinigai užima svarbią vietą. Sistemos užsakovas yra suinteresuotas kuo pigiau gauti tinkamai veikiantį produktą, o programos kūrėjas – kuo greičiau bei kokybiškiau sukurti sistemą ir atiduoti ją rinkai. Pritaikius pakartotinį panaudojimą galima sėkmingai pasiekti teigiamą rezultatą, tai yra, greičiau ir pigiau sukurti norimą informacinę sistemą. Taip pat, palyginę kelias sistemas pastebime, kad jose yra panašių dalių – komponentų, pavyzdžiui, straipsnių rašymas ir pateikimas, vartotojo užsiregistravimas, naujienų apžvalga, dinaminis navigacinis meniu ir pan.

Taigi, mūsų darbo pagrindinis tikslas - išanalizuoti komponentinio kūrimo metodus bei juos pritaikyti komponentinės interneto taikomųjų programų sistemos (karkaso) kūrime. Norint pasiekti šį tikslą, buvo įgyvendinti šie uždaviniai:

- Išanalizuota probleminė sritis, rinkos poreikiai ir komponentinių interneto sistemų kūrimo karkaso įgyvendinimo galimybės.
- Išanalizuotos panašios, jau sukurtos panašios sistemos bei atrinktos jų geros ir blogos savybės.
- Išanalizuoti komponentinių sistemų kūrimo metodai: komponentais pagrįstos programinės įrangos inžinerija (CBSE) ir šablonais paremtos programinės įrangos kūrimo architektūra (POSA).
- Nustatyti reikalavimai sistemai, rizikos faktoriai bei kokybės kriterijai.
- Nustatyti sistemos panaudojimo atvejai ir aprašytos jų specifikacijos bei nubrėžtos sekų ir veiklos diagramos kiekvienam panaudojimo atvejo, siekiant tiksliau aprašyti sistemos funkcijas.
- Sudaryta loginė sistemos architektūra, atsižvelgiant į literatūroje aprašytų metodų principus.
- Suprojektuota karkaso klasių diagrama ir priklausomybės tarp jų.

- Remiantis analizės ir projekto išvadomis, realizuotas komponentinių interneto sistemų kūrimo karkasas
- Karkasas palygintas su kitomis panašiomis sistemomis, siekiant išryškinti sukurtos sistemos savybes ir privalumus, bei įvertintas sistemos naudingumas, remiantis kokybės kriterijais ir patenkintais reikalavimais.

Kaip jau minėta, komponentinių interneto sistemų karkaso kūrimas paremtas CBSE ir POSA metodais, pasitelkus jų principus ir privalumus.

Pirmojo metodo tikslas - sukurti iš komponentų sudaryta sistemą; jos komponentus būtų galima pakartotinai panaudoti bei pakeisti kitais nesutrikdant bendro sistemos darbo. Remiantis šiuo metodu, programinės įrangos realizavimas paremtas:

- atskirų dalių (komponentų) suliejimu į vieną veikiančią sistemą;
- atskirų dalių pakartotiniu panaudojimu;
- sistemos palaikymu ir atnaujinimu, kuris būtų vykdomas pakeičiant ar modifikuojant atskiras dalis nesutrikdant bendro sistemos darbo.

Antrojo metodo pagrindinis tikslas – padėti kūrėjams išspręsti vis pasikartojančias problemas programinės įrangos vystyme. Vienas labiausiai paplitusių šablono apibrėžimų - šablonas programinės įrangos architektūroje apibūdina tam tikras specifiniame modeliavimo kontekste pasikartojančias problemas ir pristato bendras gaires šioms problemoms spręsti. Sprendimo gairės yra specifikuojamos nusakant jas sudarančius komponentus, jų specifikacijas, ryšius ir tarpusavio bendradarbiavimą.

Remiantis šių metodų principais buvo sukurtas karkasas ir jį sudarantys komponentai, kurie palengvina interneto sistemų kūrimą. Komponentai su nedideliais pakeitimais ar be jų, bus lengvai integruojami į kitas sistemas. Tokiu būdu sutaupysime ir laiko – nereikės iš naujo kurti viso komponento, ir pinigų – sumažės programavimo išlaidos.

Interneto sistemų kūrimo problemos ir komponentų pritaikymas šioms problemoms spręsti iš dalies aprašytas straipsnyje “Komponentinės architektūros pritaikymas PHP aplinkoje, internetines sistemas kuriančioje įmonėje” [13], kuris 2005 metais perskaitytas 10-je tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Informacinės technologijos“ (1 priedas). Straipsnyje kalbama apie komponentinės architektūros principus, pateikiama komponento samprata, analizuojamos problemos, kurios gali iškilti kuriant interneto sistemas, tokios kaip, sudėtingas įgyvendinimas ir neefektyvus jau sukurtų sistemų naudojimas. Šių problemų sprendimui siūloma komponentinė architektūra PHP platformai. Straipsnyje apžvelgiami komponentinės architektūros privalumai bei trūkumai, įgyvendinti sprendimai, taikymo galimybės ir pavyzdžiai.

Kadangi darbas yra kolektyvinis, kiekviena dalis turi savo autorinį indėlį. Darbo struktūrą sudaro šios pagrindinės dalys:

I. KOMPONENTINIŲ INTERNETO SISTEMŲ KŪRIMO METODŲ ANALIZĖ

1. Literatūroje esančių komponentinio kūrimo metodų analizė

a) Komponentais grįstos programinės įrangos inžinerija (Viktorija Petrošiūtė)

b) Šablonais grįstos programinės įrangos architektūra (Algirdas Simutis)

2. Panašių interneto sistemų kūrimo karkasų analizė (Viktorija Petrošiūtė)

3. Sistemos architektūros ir galimų įgyvendinimo priemonių analizė (Algirdas Simutis, Viktorija Petrošiūtė)

4. Kompiuterizuojamos karkaso funkcijos (Algirdas Simutis)

5. Reikalavimai duomenims ir nefunkciniai reikalavimai ir apribojimai (Viktorija Petrošiūtė)

6. Rizikos faktorių analizė (Viktorija Petrošiūtė)

7. Kokybės kriterijų nustatymas (Algirdas Simutis)

8. Analizės išvados (Algirdas Simutis, Viktorija Petrošiūtė)

II. KOMPONENTINIŲ INTERNETO SISTEMŲ KŪRIMO KARKASO REIKALAVIMAI (Algirdas Simutis, Viktorija Petrošiūtė)

III. KOMPONENTINIŲ INTERNETO SISTEMŲ KŪRIMO KARKASO PROJEKTAS

1. Komponentų karkaso pagrindimas ir esmės išdėstymas (Viktorija Petrošiūtė)

2. Sistemos architektūra – statinės struktūros modelis (Algirdas Simutis, Viktorija Petrošiūtė)

3. Detalus komponentų karkaso projektas (Algirdas Simutis, Viktorija Petrošiūtė)

4. Vidiniai komponentų standartai ir failų struktūra (Algirdas Simutis)

5. Duomenų bazės schema (Viktorija Petrošiūtė)

6. Realizacijos modelis (Algirdas Simutis)

IV. KOMPONENTINIŲ INTERNETO SISTEMŲ KŪRIMO KARKASO REALIZAVIMAS

1. Komponentinių interneto sistemų kūrimo karkaso realizavimas (Algirdas Simutis, Viktorija Petrošiūtė)

2. Atskiri komponentai:

a) Naujienų komponentas (Viktorija Petrošiūtė)

b) Komponentų valdymo komponentas (Viktorija Petrošiūtė)

c) Vartotojų valdymo komponentas (Algirdas Simutis)

d) Puslapio viršutinės dalies komponentas (Algirdas Simutis)

e) Puslapio apatinės dalies komponentas (Algirdas Simutis)

f) Meniu komponentas (Viktorija Petrošiūtė)

V IŠVADOS (Algirdas Simutis, Viktorija Petrošiūtė)

1. KOMPONENTINIŲ INTERNETO SISTEMŲ KŪRIMO METODŲ ANALIZĖ

1.1. Analizės tikslai ir uždaviniai

Norint tinkamai pasiruošti sistemos projektavimo etapui, reikia kruopščiai išanalizuoti probleminę sritį, rinkos poreikius ir sistemos įgyvendinimo galimybes. Siekdami atskleisti sistemos reikalingumą bei įvertinti karkaso sukūrimo galimybes, buvo iškelti pagrindiniai analizės tikslai bei uždaviniai šiems tikslams pasiekti.

Tikslai:

- Išanalizuoti esamus komponentinius interneto svetainių kūrimo karkasus, paremtus PHP technologijomis.
- Išanalizuoti komponentinių sistemų kūrimo metodus ir juos pritaikyti komponentų karkasui kurti.
- Apibrėžti savo sistemos tikslus ir siekiamus kriterijus.

Uždaviniai:

- Surinkti kuo daugiau informacijos apie esamus komponentinių interneto sistemų kūrimo karkasus, paremtus PHP technologijomis.
- Išsiaiškinti jų gerąsias ir blogąsias savybes.
- Pagal išsikeltus kriterijus atlikti sukurtų produktų savybių lyginamąją analizę.
- Surinkti reikalingą informaciją apie komponentinių sistemų įgyvendinimo specifiką bei pakartotinio panaudojimo principus.
- Nustatyti reikalavimus vartotojams, sistemos veikimui, duomenims, bei suderinamumui su kitomis sistemomis.
- Išsiaiškinti kūrimo rizikos faktorius bei nustatyti kokybės kriterijus, pagal kuriuos bus vertinamas jau sukurtos sistemos naudingumas ir tinkamas veikimas.

1.2. Tyrimo sritis, objektas ir problema

Kuriant naują sistemą gali iškilti įvairiausių problemų. Aktualiausia iš jų – neefektyvus darbo jėgos panaudojimas: nors dalis naujosios sistemos jau buvo realizuota ankstesniuose projektuose, tačiau ji kuriama iš naujo. Taip pat nesutampa programinės įrangos kūrėjų ir jos užsakovų interesai: kūrėjai nori gauti kuo didesnį pelną ir sumažinti kaštus bei laiką skirtus sistemos realizavimui, o klientai reikalauja kuo geresnės kokybės ir sistemos pritaikymo jų individualiems poreikiams. Dėl rinkos evoliucijos pastebimas greitas programinių produktų sudėtingumo ir apimčių augimas.

Didėjant produktų sudėtingumui išryškėja šios tipinės problemos:

- skirtingų sistemų dalims kurti pritaikomi tie patys veikimo principai, tačiau jos kuriamos iš naujo;
- identiškos savybės, skirtingose sistemose, elgiasi skirtingai;
- sistemų atnaujinimas reikalauja daug pastangų ir išteklių.

Norint išspręsti šias problemas, reikia suprojektuoti ir realizuoti karkasą, kurio pagrindu būtų kuriamos komponentinės interneto sistemos.

Taigi, tyrimo sritis yra komponentiniai interneto sistemų kūrimo metodai, tyrimo objektas - komponentų karkasas.

1.3. Organizacijos, kuriančios interneto sistemas, veiklos analizė

Interneto sistemų kūrimo karkasas bus realizuojamas organizacijoje, kurios pagrindinė veiklos funkcija - interneto sistemų kūrimas. Šią funkciją sudaro procesai:

- Analizė;
- Projektavimas;
- Realizavimas;
- Testavimas;
- Palaikymas.

Kiekvieną šių procesų skaidome į smulkesnius procesus.

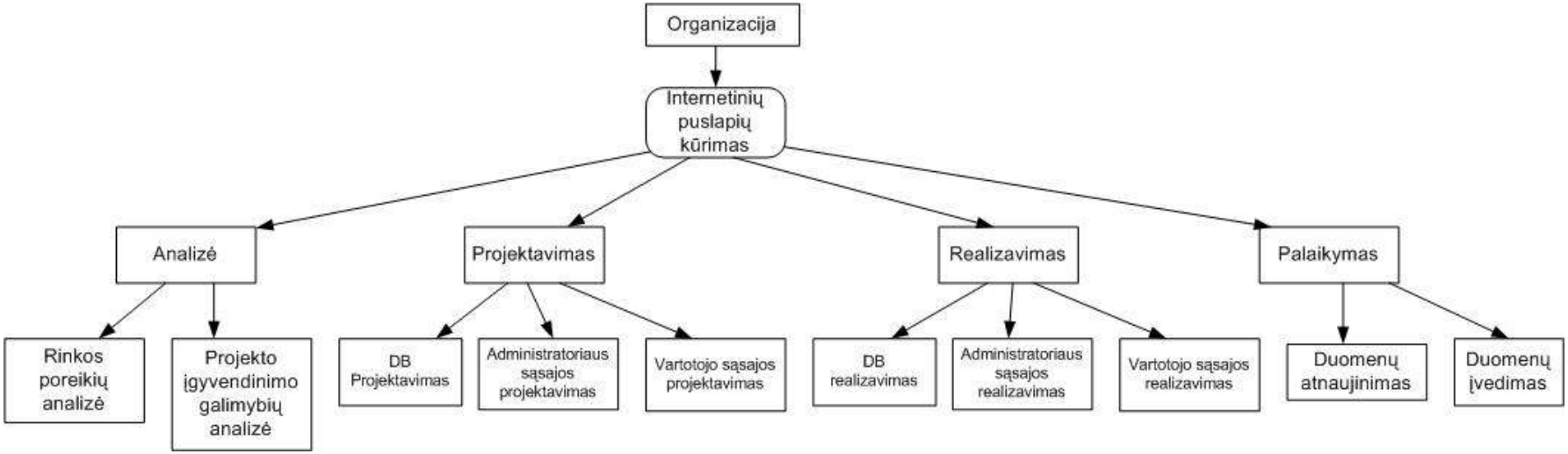
Analizės etape išnagrinėjama rinka, naujos sistemos poreikiai ir galimybės jai įgyvendinti. Jei analizės metu išaiškėja, kad sistemos kūrimas naudingas ir organizacijai, ir klientams, sistema pradedama projektuoti.

Projektavimo etape nustatomos vartotojo ir administratoriaus funkcijos, sukuriami jos panaudojimo atvejai, bei kiekvieno panaudojimo atvejo veiklos bei sekų modeliai, kad būtų lengviau įsivaizduoti, kaip sistema turėtų veikti, ir nustatomi sistemos reikalavimai bei funkcijos. Vėliau suprojektuojama duomenų bazės schema, vartotojo sąsaja ir, galiausiai, sukuriamas dizainas.

Realizacijos etape įgyvendinamos projektavimo etape nustatytos sistemos funkcijos. Tai daugiausiai pinigų ir laiko reikalaujantis etapas.

Sukūrus sistemą į ją suvedami duomenys ir lygiagrečiai vykdomas testavimas. Vėliau vyksta duomenų atnaujinimas ir atliekami reikalingi sistemos pakeitimai.

Šių etapų hierarchija pateikta 1.1 paveiksle pavaizduotame organizacijos veiklos procesų modelyje.



1.1 pav. Organizacijos veiklos procesų modelis

1.4. Vartotojų analizė

1.4.1. Vartotojų aibė tipai ir savybės

Kalbant apie kuriamą sistemą, galima išskirti tiesioginį ir netiesioginį jos vartojimo tipą: pirmasis - kai naudojamasi pačiu karkasu, norint sukurti interneto sistemą; antrasis – komponentų karkaso pagrindu sukurtos sistemos vartojimas.

Įsigilinus į šiuos vartojimo tipus, suformuojama vartotojų aibė:

- įmonės ir organizacijos, kurios pristato savo veiklą;
- pavieniai asmenys, pateikiantys informaciją apie save, dalyvaujantys diskusijose bei forumuose ir panašiai;
- sistemų administratoriai;

Apibendrinus šios aibės vartotojus, išskiriame du jos tipus:

- Sistemos administratoriai – tai asmenys, kurie kuria sistemas pasitelkdami komponentinių interneto sistemų kūrimo karkasą.
- Sistema besinaudojantys vartotojai – naudojantys jau sukurtas interneto sistemas, pateikiantys informaciją apie save, skaitantys pateiktas naujienas ir panašiai.

1.4.2. Vartotojų tikslai ir problemos

Kiekvienas sistemos administratorius naudojantis komponentų karkasą yra suinteresuotas, jog karkaso sąsaja būtų intuityvi, paprasta ir lengvai naudojama. Tuo tarpu klientas ieškantis tinklalapio pagal jį dominančią sritį, norės naršyti tik informatyviame, patogiai išdėstytame puslapyje.

Taigi, sistemos administratoriaus tikslas - surinkti iš komponentų sistemą pritraukiančią klientą ir intuityvią jam naudotis. Kliento pagrindinis tikslas yra surasti patikimą, informatyvią ir patogią naudotis sistemą.

Norint, nustatyti dabartinę interneto sistemų kūrimo situaciją, tikslinga apibrėžti šia veikla užsiimantiems vartotojams iškylančias problemas. Jos aprašytos 1.1 lentelėje.

1.1 lentelė. Interneto sistemų kūrimo veikla užsiimančių vartotojų problemos

Problema	Kas vyksta dabar
Interneto sistemų kūrimas yra lėtas procesas.	Skirtingų sistemų dalims kurti pritaikomi tie patys veikimo principai, tačiau jie kuriami iš naujo. Tai užima daugiau laiko.
Interneto sistemos dizaino pakeitimas reikalauja daug pastangų.	Norint pakeisti dizainą, iš naujo perrašomi šablonų ir stilių failai, kurie dažniausiai būna nesistemingi.
Atliekant pakeitimus interneto sistemoje sutrinka sistemos funkcionalumas.	Atskiros sistemos dalys yra glaudžiai susijusios viena su kita, todėl vienos dalies pakeitimas sutrikdo dalies ar net visos sistemos darbą.
Paprastam vartotojui sudėtinga sukurti naują interneto sistemą. Tai reikalauja papildomų žinių	Norėdamas sukurti savo interneto sistemą vartotojas mažiausiai turi žinoti HTML.

Siekdami išspręsti 1.1 lentelėje aprašytas problemas realizuosime komponentų karkasą, kuris bus kaip pagrindas interneto sistemoms kurti. Jame bus integruoti dalis pagrindinių interneto sistemas sudarančių komponentų, kurie bus panaudojami naujai sistemai surinkti. Šiuo komponentų karkasu galės naudotis tiek paprastas, nieko nežinantis apie interneto sistemų specifiką vartotojas, tiek patyręs interneto sistemų kūrėjas. Tai palengvins kūrimo procesą, sutaupys laiko ir pinigų.

1.5. Literatūroje pateikiamų komponentinio kūrimo metodų analizė

Literatūros šaltiniuose aprašyti du metodai panašioms problemoms spręsti: komponentais pagrįstos programinės įrangos inžinerija (CBSE – Component-Based Software Engineering) ir į šablonus orientuotos programinės įrangos architektūra (POSA – Pattern Oriented Software Architecture).

1.5.1. Komponentais pagrįstos programinės įrangos inžinerija (CBSE – Component Based Software Engineering)

Komponentinės programinės įrangos inžinerija apima pagrindinius pakartotinio naudojimo ir komponentų principus.

Pagrindinis metodo tikslas [2] – sukurti sistemą, kuri būtų sudaryta iš komponentų; jos komponentus būtų galima pakartotinai panaudoti bei pakeisti kitais nesutrikdant bendro sistemos darbo.

Remiantis šiuo metodu, programinės įrangos realizavimas paremtas:

- atskirų dalių (komponentų) suliejimu į vieną veikiančią sistemą;
- atskirų dalių pakartotiniu panaudojimu;
- sistemos palaikymu ir atnaujinimu, kuris vykdomas pakeičiant ar modifikuojant šias dalis nesutrikdant bendro sistemos darbo.

Didėjant realizuojamų sistemų apimtims, jų kūrėjai tradiciškai susiduria su sistemų sudėtingumo ir sistemos priklausomybės nuo tam tikros išorinės sistemos sunkumais. Dažniausiai jie susitelkia į vienos sistemos kūrimą, jos užbaigimo terminus ir tam skirtą biudžetą, ignoruodami bendrus, nuolat pasikartojančius sistemų vystymo poreikius. Tai veda prie daugumos projektų žlugimo dėl laiku neatlikto darbo ar pinigų trūkumo, o galiausiai dėl kokybės reikalavimų neatitikimo arba per didelių palaikymo kaštų.

Norint išvengti sudėtingumo, sistemas turi būti kuriamos taip, kad atsiradus poreikiui jų dalis būtų galima greitai pakeisti. Nesilaikant šio principo sistemą gali tekti perrašyti iš naujo.

Realizuojant sistemą iš komponentų, ji bus gaunama sujungiant jau anksčiau sukurtus ir paruoštus integravimui komponentus.

Komponentai – tai programinės įrangos struktūriniai vienetai, kurie remiasi tam tikrais specifiniais principais [4] :

- Sistemą sudarančius komponentus vienija bendras standartas.
- Komponento veiklos rezultatas priklauso nuo specifikacijos, o ne nuo jo realizacijos.

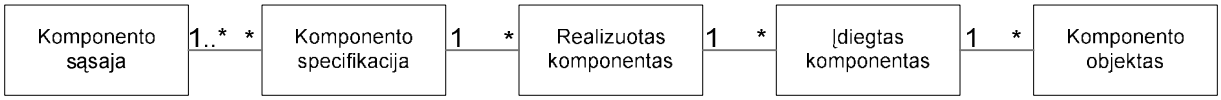
- Kiekvienas programinės įrangos komponentas turi jam unikalią identifikaciją, nepriklausomai nuo jo būsenos.

Remiantis šiais principais galime išskirti tokias komponentų savybes [1]:

- Komponentas teikia paslaugas nepriklausomai nuo to, kur jis vykdomas, ir nepriklausomai nuo jo programavimo kalbos.
- Komponentas yra nepriklausoma vykdomoji programos dalis, kuri gali susidėti iš vienos ar daugiau vykdomųjų objektų.
- Visi veiksmai vykdomi per komponento sąsają.
- Komponentų dydžiai gali skirtis nuo paprastų funkcijų iki taikomųjų sistemų.

Komponentas turi [4]:

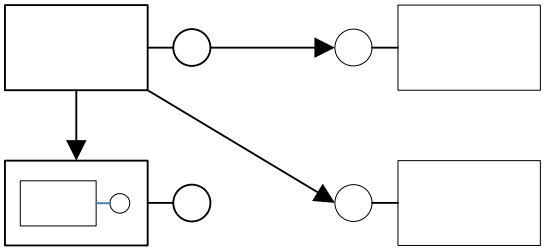
- standartą – galimybę surinkti sistemą iš atskirų komponentų lemia vieningi standartai;
- specifikaciją;
- sąsają;
- realizaciją;
- įdiegimą;
- objektą – tik komponento objektas gali atlikti veiksmus.



1.2 pav. Komponentas nuo jo specifikacijos iki objekto

1.2 paveiksle pavaizduotas komponentas nuo specifikacijos iki objekto [4]. Pagal specifikaciją kuriama komponento sąsaja. Komponentas gali turėti vieną arba daugiau sąsajų. Sukurtas komponentas įdiegiamas ir gali bendrauti su kitais komponentais. Komponento panaudojimas vyksta sukūrus vieną ar kelis jo objektus. Iš komponentų yra sudaroma sistema, kurios atskiros dalys gali būti panaudojamos iš naujo, taip pat jas galima pakeisti naujesnėmis be sistemos veikimo sutrikimų.

Grafinis komponentų ryšių atvaizdavimas leidžia suprasti, kaip glaudžiai yra susijusi komponentinė sistema ir numatyti sistemos modifikavimo pasekmes. Galimi ryšiai tarp komponentų pavaizduoti 1.3 paveiksle. [13]



1.3 pav. Ryšiai tarp komponentų

Komponentai tarpusavyje gali sietis:

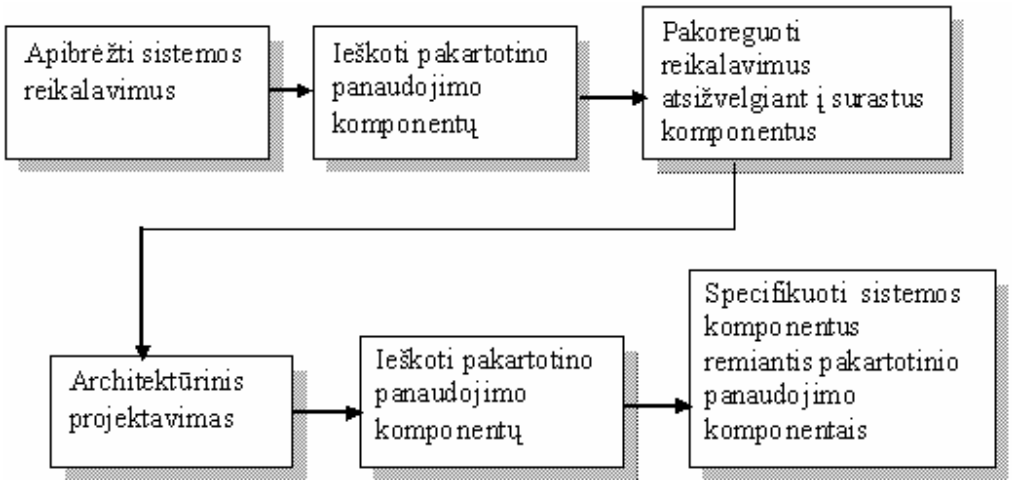
- sąsaja su sąsaja;
- komponentas su komponentu;
- komponentas su sąsaja;
- komponentas gali būti kito komponento dalimi.

CBSE apjungia komponentų kūrimą ir sistemos kūrimą panaudojant komponentus. Kuriant komponentus galima įtraukti ir kitus komponentus, tačiau didžiausias dėmesys skiriamas pakartotiniam panaudojimui: komponentai kuriami ne tik tam, kad juos panaudotume konkrečių sistemų kūrimui, bet kad juos būtų galima pakartotinai panaudoti kitose panašiose sistemose. Komponentas turi būti lengvai pritaikomas ir pakeičiamas kitu. Komponento sąsaja turi būti paprasta ir atskirta nuo įgyvendinimo.

Pagrindinė komponentinio kūrimo problema yra reikalavimų supratimas ir jų suderinimas su sistemą sudarančių komponentų parinkimu. Problema iškyla todėl, kad šis procesas reikalauja vienu metu priimti kelis svarbius sprendimus. Jei pirma nustatysime reikalavimus ir tik po to kursime programą, labai didelė tikimybė, kad ne visi būtini reikalavimai bus apgalvoti. Jei pirmiausia parinksime komponentus, o vėliau apgalvosime būtinus reikalavimus, vėl iškyla rizika, kad sukurta sistema neatitiks visų reikalavimų.

Kuriant sistemą iš komponentų pagrindinis dėmesys skiriamas pakartotinai panaudojamų esybių ir ryšių tarp jų identifikavimui pradedant nuo sistemos reikalavimų. Pirminį modeliavimo procesą sudaro du esminiai žingsniai: sistemos architektūros specifikavimas funkcinių komponentų bei jų tarpusavio sąveikos atžvilgiu (taip matomas sistemos loginis vaizdas) ir sistemos architektūros, kurią sudaro fiziniai komponentai, specifikavimas.

1.4 paveiksle pavaizduotas sistemos sudarytos iš komponentų kūrimo ciklas [1].



1.4 pav. Kūrimo ciklas su pakartotinai panaudojamais komponentais

Tokios sistemos kūrimą sudaro šie procesai [2]:

- Surasti komponentus, kurie gali būti panaudoti sistemai kurti.
- Išrinkti komponentus, kurie atitinka sistemai keliamus reikalavimus.
- Kaip alternatyvą, sukurti komponentus, kurie bus panaudoti sistemos realizavimui.
- Pritaikyti išrinktus komponentus taip, kad jie atitiktų egzistuojančio komponento modelį arba reikalavimų specifikacijas.
- Išdėstyti komponentus naudojant komponentų karkasą.
- Senesnius komponentus pakeisti naujesnėmis komponentų versijomis. Tai komponentų palaikymo etapas, kai ištaisomi esami trūkumai, ar pridedamos naujos funkcijos.

Komponentų panaudojimo privalumai [2]:

- Efektyvus produktų sudėtingumo valdymas – sistema išskaidoma į mažesnius komponentus, kurie veikia nepriklausomai vienas nuo kito;
- Geresnė kokybė;
- Platesnis panaudojimo diapazonas.

Komponentų panaudojimo trūkumai:

- Sukurti pakartotinai panaudojamą stabilios būsenos komponentą užima daugiau laiko ir pastangų nei sukurti komponentą pritaikytą dirbti vienoje sistemoje.
- Sunku sukurti bendrus reikalavimus komponentams, kurie vėliau bus naudojami įvairiose sistemose.
- Palaikymo kaštai gali būti labai dideli, nes komponentas turi palaikyti skirtingų sistemų skirtingus reikalavimus.

- Sistemos ir komponentai turi skirtingus gyvavimo ciklus ir reikalavimus, todėl yra rizika, kad komponentas negalės visiškai išpildyti sistemos reikalavimų.

1.5.2. Programinės įrangos orientuotos į šablonus architektūra (POSA – Pattern-oriented software architecture) [11, 12]

Dažniausiai sistemų kūrėjai linkę to paties tipo problemai galvoti vis naują sprendimą vietoj to, kad prisimintų anksčiau spręstas panašias problemas ir pritaikytų esminius senų sprendimų aspektus. Ankstesnių sprendimų pritaikymas naujoms problemoms spręsti veda prie pakartotinai naudojamų šablonų, kurių pradininkas yra Kristoferis Aleksandras (Christopher Alexander).

Pagrindinis šablonų tikslas – padėti kūrėjams išspręsti vis pasikartojančias problemas programinės įrangos kūrime. Vienas labiausiai paplitusių šablono apibrėžimų yra Bušmano (Buschmann): šablonas programinės įrangos architektūroje apibūdina tam tikras pasikartojančias problemas, kurios iškyla specifiniame modeliavimo kontekste ir pristato bendras gaires tų problemų sprendimui. Sprendimo gairės yra specifikuojamos nusakant jas sudarančius komponentus, jų specifikacijas, ryšius ir tarpusavio bendradarbiavimą.

Šablonų kūrėjai apibrėžė tris šablonų tipus:

- Architektūriniai šablonai – išreiškia pagrindinę struktūrinę organizaciją ar struktūrą programinės įrangos sistemoms. Jie tiekia jau paruoštas posistemes, specifikuoja jų atsakomybes, taip pat apima taisykles ryšiams tarp jų formuoti.
- Modeliavimo šablonai – pateikia struktūrą sistemos posistemėms, komponentams bei ryšiams tarp jų tobulinti. Tai paprastai aprašo tarpusavyje bendraujančių komponentų, kurie sprendžia pagrindinę modeliavimo problemą tam tikrame kontekste, besikartojančią struktūrą.
- Idiomos – tai žemo lygio šablonai skirti programavimo kalbai. Idiomos nusako, kaip įgyvendinti atskiras problemas komponentų arba ryšių tarp jų atžvilgiu remiantis duotos programavimo kalbos savybėmis.

Skirtingi šablonų tipai veikia tik tam tikruose abstrakcijos lygiuose. Architektūriniai šablonai yra tarsi aukšto lygio strategijos, kurios sieja didelės apimties komponentus ir globalias sistemų savybes bei mechanizmus. Pačios sąsajos veikia visą pamatinę programinės įrangos sistemos struktūrą ir organizaciją. Modeliavimo šablonai yra vidutinės apimties taktikos, kurios išplėtoja kai kurių esybių struktūrą ir elgesį bei ryšius tarp jų. Jos neįtakoja visos sistemos struktūros, tačiau apibrėžia posistemių ir komponentų mikroarchitektūras. Idiomos yra ypač specifiniai programavimo būdai, kurie išpildo žemo lygio vidines ir išorines komponentų struktūros bei elgesio savybes.

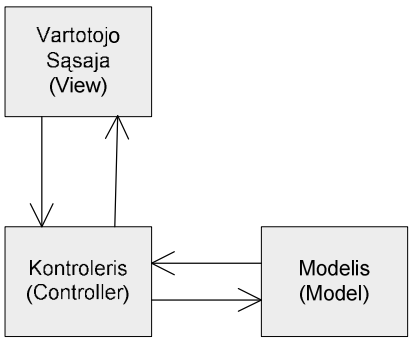
Vienas iš žymiausių architektūrinių šablonų yra Model-View-Controller (MVC), kuris buvo išvystytas naudojant Smalltalk programavimo aplinką vartotojo sąsajoms kurti. Šis šablonas yra naudojamas kuriant interaktyvias sistemas. Pagrindinis jo tikslas – atskirti sistemos objektą (Model) nuo būdo, kuriuo jis yra atvaizduojamas vartotojui (View), ir nuo valdymo (Controller).

Modelis (Model) žino apie visus duomenis, kuriuos reikia atvaizduoti bei operacijas, kurias reikia atlikti, kad modelis būtų vienaip ar kitaip transformuotas. Kita vertus, jis nieko nežino apie vartotojo sąsają, būdus kaip duomenys bus atvaizduoti bei grafinės vartotojo sąsajos veiksmus, kurie naudojami manipuliuoti duomenimis. Priėjimo prie duomenų ir manipuliavimo jais metodai visiškai nepriklauso nuo grafinės vartotojo sąsajos.

Vartotojo sąsaja (View) siejasi su kontroleriu (Controller) ir yra skirta iš modelio gautai informacijai atvaizduoti.

Kontroleris žino metodus, kuriais vartotojas manipuliuoja modelio duomenimis. Pavyzdžiui, grafinėje vartotojo sąsajoje valdymo objektas gaus pelės ar klaviatūros paspaudimus kaip tam tikrus įvedimo duomenis, kuriuos jis pavers į tokius duomenis, kuriuos modelio objektas supras.

1.5 paveiksle matome grafiškai atvaizduotą MVC šabloną.



1.5 pav. MVC (Model-View-Controller) šablonas

1.5.3. Literatūroje pateikiamų komponentinio kūrimo metodų apibendrinimas

Palyginus aukščiau aprašytus metodus pagal 1.2 lentelėje nurodytus kriterijus, matosi, kad jie padeda spręsti tas pačias problemas. Tikslinga naudoti abu metodus kartu, kadangi jie duoda sinergetinį efektą (sustiprina vienas kito veikimą).

1.2 lentelė. Metodų apibendrinimas

Metodas / Kriterijus	CBSE	POSA
Sistemos realizavimo laiko sutrumpinimas ir kaštų sumažinimas	padeda	padeda
Produktų sudėtingumo valdymas	padeda	padeda
Aukštesnė produkto kokybė	padeda	padeda
Palengvina produkto tolimesnį palaikymą	padeda	padeda
Padidintas patikimumas	padeda	padeda
Pakartotinis panaudojimas	padeda	padeda

Bus kuriamas karkasas ir įvairios paskirties komponentai, kuriuos bus galima naudoti kuriant interneto sistemas. Kadangi mes norime sukurti patikimą, kokybišką ir lengvai valdomą sistemą, pasirinkome CBSE metodą, nes jis nagrinėja visas mums rūpimas problemas ir pagrįstas pakartotinio panaudojimo ir komponentų principais. CBSE sudėtingų sistemų valdymui siūlo principą „skaldyk ir valdyk“. Problema suskaidoma į mažesnes ir kiekviena iš jų sprendžiama atskirai, o vėliau visi sprendimai apjungiami į visumą. Didžiausias uždavinys yra sujungti funkcijas ir su jomis susijusius duomenis į vieną veikiančią sistemą. Šio uždavinio sprendimui naudojamos komponentų savybės.

Kalbant apie komponentų savybes, viena iš pagrindinių savybių – komponentas privalo būti lengvai pakeičiamas kitu – esamo komponento atnaujinta versija arba visiškai nauju, tą pačią sąsają palaikančiu komponentu. Pritaikius komponentų pakartotinį naudojimą, sistema gali būti lengvai išplečiami naujomis funkcijomis. Tokie komponentai - patikimi ir kokybiški, nes yra patikrinti jau anksčiau sukurtose sistemose.

Komponentinis kūrimas nuo kitų metodų skiriasi tuo, kad specifikavimas yra atskiriamas nuo integravimo ir padalinamas į sąsajas (interface). Komponentų specifikacijos padalinimas į vieną ar daugiau sąsajų reiškia, kad tarpkomponentinės priklausomybės gali būti apribojamos individualiais ryšiais tarp susijusių komponentų, o ne vieno komponento ryšiu su visa sistema. Tai sumažina komponentų atnaujinimo įtaką sistemai. Galima vieną komponentą pakeisti kitu su skirtinga specifikacija, bet turinčiu tą pačią sąsają.

Kuriant interaktyvią sistemą rekomenduoma naudoti architektūrinį MVC šabloną. Jis atskiria modelį, vartotojo sąsają ir sistemą vienus nuo kitų, todėl palengvėja tolimesnis sistemos palaikymas, nes norint pakeisti vieną iš sistemos lygmenų, tai galima atlikti neliečiant likusių.

1.6. Panašių interneto sistemų kūrimo karkasų analizė

Norint nustatyti būsimo karkaso savybes, tikslinga išanalizuoti jau sukurtus panašius karkasus bei išsiaiškinti jų gerąsias ir blogąsias savybes, kad būtų galima sukurti pranašesnę sistemą. Tuo tikslu buvo išnagrinėtos panašios sistemos PHP-Nuke, Xoops, Prado ir Smart3.

1.6.1. PHP-Nuke [5]

Tai turinio valdymo sistema, kurią sudaro įvairūs komponentai, skirti interneto sistemoms kurti Sukurta naudojant PHP programavimo kalbą, bendrauja su SQL duomenų baze ir veikia Apache serveryje.

Trumpa PHP-NUKE karkaso apžvalga pateikta 1.3 lentelėje.

1.3 lentelė PHP-NUKE aprašymas

Prieiga internetu:	http://www.phpnuke.org
Reikalavimai veikimui:	- MySQL duomenų bazė; - Apache serveris; - PHP 4
Privalumai:	- lengvai keičiamos sukurto puslapio galimybės; - vartotojas neprivalo mokėti HTML ar PHP, kad galėtų sukurti savo svetainę.
Trūkumai:	- nenaudojamas objektinis programavimas; - įdiegus ne visos dalys veikia; - neatitinka XHTML standarto; - nenaudoja šablonų. - nenaudojamas MVC

1.6.2. Plečiama objektinė portalų kūrimo sistema XOOPS

Skirta įvairiems dinaminiams interneto puslapiams, weblogams ar portalams kurti. Sukurta sistema valdoma naudojant Xoops modulių administravimo sistemą.

XOOPS (eXtensible Object Oriented Portal System) [6] trumpas aprašymas pateikiamas 1.4 lentelėje.

1.4 lentelė. XOOPS aprašymas

WWW adresas:	http://www.xoops.org
Reikalavimai veikimui:	- MySQL duomenų bazė; - Apache serveris; - PHP 4
Privalumai:	- nemokama; - lengvai keičiamos sukurto puslapio galimybės; - vartotojas neprivalo mokėti HTML ar PHP, kad galėtų sukurti savo svetainę. - sistema atitinka XHTML standartus; - naudojamas objektinis programavimas; - naudojami šablonai.
Trūkumai:	- dalis modulių neveikia. - nenaudojamas MVC

1.6.3. PRADO karkasas

Tai karkasas skirtas interneto puslapiams kurti. Kuriant panaudotos idėjos iš Borland Delphi, Microsoft ASP.NET, naudojama PHP 5-ta versija.

Trumpa PRADO karkaso apžvalga pateikta karkaso [7] aprašymas pateiktas 1.5 lentelėje.

1.5 lentelė. PRADO aprašymas

WWW adresas:	http://www.xisc.org
Reikalavimai veikimui:	- MySQL duomenų bazė; - Apache serveris; - PHP 5.
Privalumai:	- nemokama; - naudojamas objektinis programavimas;
Trūkumai:	- šablonams kurti naudojami savi standartai, o ne HTML; - neintuityvus naudojimasis karkasu;

1.6.4. SMART3 karkasas

PHP karkasas komponentams kurti, kurie gali būti panaudojami ne tik interneto puslapiams. Turi du šablonų atvaizdavimo mechanizmus.

SMART3 [8] karkasas aprašytas 1.6 lentelėje.

1.6 lentelė. SMART3 aprašymas

WWW adresas:	http://www.smart3.org
Reikalavimai veikimui:	- MySQL duomenų bazė; - Apache serveris; - PHP 5.
Privalumai:	- naudojamas objektinis programavimas; - integruotos dvi šablonų valdymo sistemos.
Trūkumai:	- neintuityvus naudojimasis karkasu;

Siekiant parodyti šių sistemų savybes,

1.7 lentelėje pateikta jų analizė pagal šiuos kriterijus:

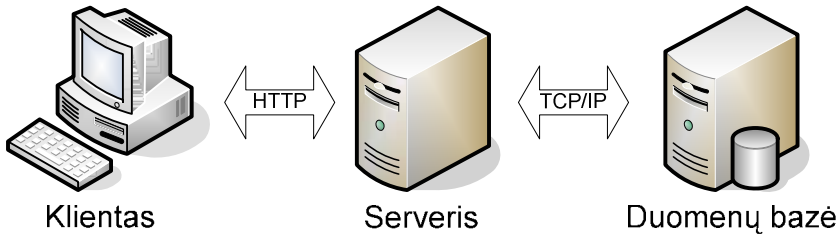
- Interneto sistema sukurta karkaso pagrindu turi būti lengvai keičiama, t.y. lengvai keičiamos atskiros sistemos dalys.
- Turi būti galimybė pačiam vartotojui kurti naujus komponentus ir juos integruoti į sistemą.
- Sistema turi turėti kelis, mažiausiai paprasto vartotojo ir administratoriaus lygius.
- Turi naudoti objektinį programavimą, kuris leidžia kurti labiau struktūrizuotą programos kodą, kas palengvina tolimesnį sistemos plėtojimą ir modifikavimą.
- Sistema turi būti sukurta naudojant PHP5. PHP5 nuo ankstesnių versijų skiriasi tuo, kad ji yra objektinė programavimo kalba.
- Turi naudoti MVC architektūrinį šabloną. Šis šablonas atskiria sistemos logiką (Model), vartotojo sąsają (View) ir valdymą (Controller) vienus nuo kitų, tai palengvina tolimesnį sistemos palaikymą.
- Karkasas turi laikytis XHTML standarto, norint, kad visos naršyklės teisingai atvaizduotų sukurtą sistemą.
- Tūri būti naudojami šablonai (templates). Tokiu būdu atskiriamas programavimas nuo atvaizdavimo.
- Sistemos įdiegimas turi būti lengvas ir intuityvus.
- Įdiegus naują sistemą, visos jos dalys turi veikti tvarkingai

1.7 lentelė. Lyginamoji interneto sistemų kūrimo karkasų analizė

Sistemos savybės	PHP-NUKE	Xoops	Prado	Smart3
Lengvai keičiamos sukurtas puslapis	+	+	?	?
Galimybė pačiam kurti ir integruoti naujus komponentus	+	+	+	+
Užtikrintos vartotojų teisės	+	+	?	?
Kuriant naudojamas objektinis programavimas	-	+	+	+
Kuriant naudojama PHP 5	-	-	+	+
Naudojamas MVC architektūrinis šablonas	-	-	-	-
Sistemos duomenys atitinka XHTML standartą	-	+	-	?
Naudojami šablonai	-	+	-	+
Paprastas įdiegimas	+	+	+/-	+
Įdiegus sistemą, visos sistemos dalys funkcionuoja	-	-	+	+
Nemokama	-	+	+	+

1.7. Sistemos architektūros ir galimų įgyvendinimo priemonių variantų analizė

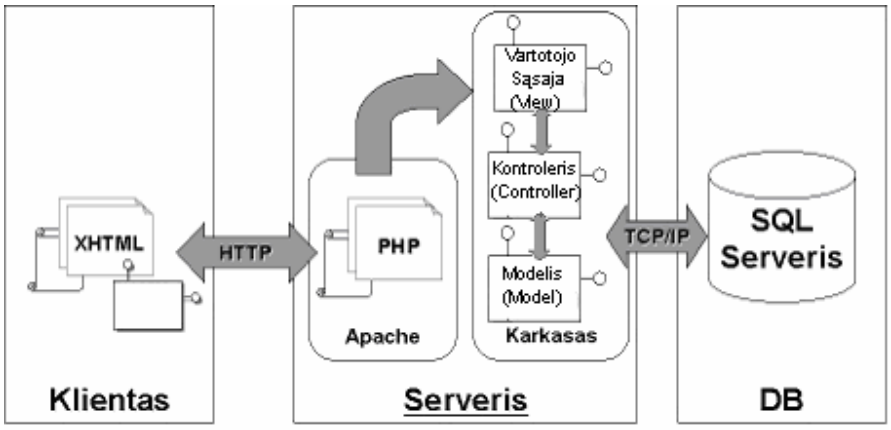
Karkasas duomenis saugos duomenų bazėje, o vartotojas juo naudosis naršyklės pagalba, todėl apibendrintą sistemos architektūrą sudaro klientas, serveris ir duomenų bazė (1.6 paveikslas).



1.6 pav. Apibendrinta sistemos architektūra

Klientas – tai interneto naršyklė (pvz. Internet Explorer, Mozilla, Netscape ir kt.). Kaip serverį ir duomenų bazę galima pasirinkti arba Microsoft IIS ir, pavyzdžiui, MSSQL, arba Apache ir, pavyzdžiui, MySQL. Naudojant pirmąjį variantą, karkasą reikėtų kurti .NET aplinkoje, antruoju atveju - PHP. Mūsų kuriamai sistemai įgyvendinti tinka abu variantai, tačiau pirmasis brangiai kainuoja, tuo tarpu antrasis yra visiškai nemokamas ir kol kas plačiau paplitęs.

Detaliau nagrinėjant sistemą, jos architektūrą galima pavaizduoti taip, kaip parodyta 1.7 paveiksle.



1.7 pav. Detalesnė būsimos sistemos architektūra

Papildomi įrankiai naudojami sistemai kurti:

- PEAR, kuris įdiegiamas kartu su PHP ir tampa beveik neatsiejama jo dalimi;
- Smarty – labiausiai paplitusi PHP šablonų vaizdavimo sistema.

1.8. Darbo tikslas ir siekiami privalumai

Šio darbo tikslas – išanalizuoti komponentinio kūrimo metodus bei juos pritaikyti komponentinių interneto sistemų kūrimo karkasui realizuoti. Sukurtas karkasas turėtų būti lankstus naudojimo atžvilgiu (turi tenkinti tiek patyrusio, tiek naujo vartotojo poreikius), lengvai suprantamas, intuityvus, lengvai modifikuojamas, palaikyti papildomų komponentų įdiegimą ar esamų pašalinimą nesutrikdant bendro interneto sistemos funkcionalumo.

1.8 lentelėje pateikėmi siekiamos sistemos privalumai lyginant su dabartine situacija.

1.8 lentelė. Siekiamos sistemos privalumai lyginant su dabartine situacija

Siekiamos sistemos privalumai	Kaip yra dabar
Įdiegus sistemą tikimės, kad ženkliai sumažės laiko ir pinigų sąnaudos skirtos vienai interneto sistemai kurti.	Kiekvieną kartą sistema kuriama iš naujo.
Sistema sudaryta iš komponentų, kuriuos bus galima pakartotinai naudoti.	Iš dalies sistema kuriama iš atskirų dalių, kurios glaudžiai susietos viena su kita. Dėl šios priežasties išėmus vieną iš dalių sutrinka susijusių dalių veikimas.
Komponentams kurti pritaikyti originalūs vidiniai standartai.	Atskiros sistemos dalys nestandartizuotos, todėl norint jas pritaikyti kitose sistemose reikia atlikti daug pakeitimų.
Susisteminti šablonų ir stilių failai	Norint pakeisti dizainą, iš naujo perrašomi šablonų ir stilių failai, kurie dažniausiai yra nesusisteminti, todėl pakeitimas užima daugiau laiko.

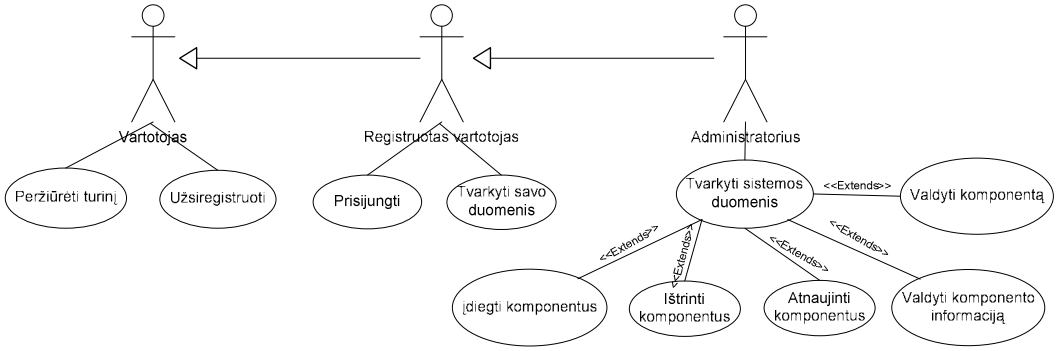
Manome, kad karkaso sukūrimas išspręs dabartines aprašytas problemas ir sistema galės naudotis interneto sistemų kūrimo veikla užsiimančios įmonės ir vartotojai.

1.9. Kompiuterizuojamos sistemos funkcijos

Sistemoje išskirti 3 tipų aktoriai:

- Paprastas vartotojas – gali peržiūrėti sistemos turinį ir užsiregistruoti.
- Registruotas vartotojas – paveldi paprasto vartotojo funkcijas. Turi galimybę prisijungti prie sistemos ir tvarkyti duomenis apie save.
- Sistemos administratorius – paveldi registruoto ir paprasto vartotojo funkcijas, taip pat gali peržiūrėti visus sistemos vartotojus, tvarkyti vartotojus bei komponentus ir jų teikiamą informaciją.

Šie tipai yra susiję tarpusavyje, todėl juos galima pavaizduoti hierarchine struktūra, pradedant paprastu vartotoju ir baigiant sistemos administratoriumi. Sistemos panaudojimo atvejai vartotojų atžvilgiu pavaizduoti 1.8 paveiksle.



1.8 pav. Sistemos panaudojimo atvejai

1.9.1. Pradinės panaudojimo atvejų specifikacijos

1.9 – 1.17 lentelėse pateiktos pradinės sistemos panaudojimo atvejų, pavaizduotų 1.8 paveiksle, specifikacijos, siekiant kuo tiksliau apibrėžti kompiuterizuojamas sistemos funkcijas.

1.9 lentelė. Panaudojimo atvejo „Peržiūrėti turinį“ pradinė specifikacija

Tikslas	Peržiūrėti sistemoje pateiktą informaciją
Aktorius	Vartotojas
Ryšiai su kitais panaudojimo atvejais	-
Nefunkciniai reikalavimai	Patogi sąsaja
Prieš sąlygos	-
Sužadinimo sąlyga	Vartotojas aktyvuoja sistemą
Po sąlyga	-
Pagrindinis scenarijus	-
Alternatyvūs scenarijai	-

1.10 lentelė. Panaudojimo atvejo „Užsiregistruoti“ pradinė specifikacija

Tikslas	Užsiregistruoti vartotojui
Aktorius	Vartotojas
Ryšiai su kitais panaudojimo atvejais	Vartotojas inicijuoja registracijos formą
Nefunkciniai reikalavimai	Saugiai perduoti vartotojo informaciją
Prieš sąlygos	Vartotojas neužsiregistravęs
Sužadinimo sąlyga	Vartotojas kreipiasi į sistemą
Po sąlyga	Vartotojas užregistruotas
Pagrindinis scenarijus	Vartotojas pasirenka registravimo mygtuką, įveda savo duomenis, patvirtina
Alternatyvūs scenarijai	- Vartotojo duomenys nekorektiški; - Vartotojas atšaukia registraciją.

1.11 lentelė. Panaudojimo atvejo „Prisijungti“ pradinė specifikacija

Tikslas	Prisijungti prie sistemos
Aktorius	Registruotas vartotojas
Ryšiai su kitais panaudojimo atvejais	-
Nefunkciniai reikalavimai	Užtikrinti slaptažodžio saugumą
Prieš sąlygos	Vartotojas užsiregistravęs sistemoje
Sužadinimo sąlyga	Vartotojas paspaudžia prisijungimo mygtuką
Po sąlyga	Vartotojas prijungtas
Pagrindinis scenarijus	Vartotojas įveda prisijungimo vardą, slaptažodį, paspaudžia prisijungimo mygtuką
Alternatyvūs scenarijai	- Vartotojo duomenys nekorektiški - Vartotojas neregistruotas

1.12 lentelė. Panaudojimo atvejo „Tvarkyti savo duomenis“ pradinė specifikacija

Tikslas	Pakeisti asmeninius duomenis
Aktorius	Registruotas vartotojas
Ryšiai su kitais panaudojimo atvejais	-
Nefunkciniai reikalavimai	- Patogi ir intuityvi sąsaja - Saugumas
Prieš sąlygos	Vartotojas prisijungęs prie sistemos
Sužadinimo sąlyga	Vartotojas pasirenka duomenų tvarkymo veiksmą
Po sąlyga	Vartotojo duomenys atnaujinti
Pagrindinis scenarijus	Vartotojui parodomi jo esami duomenys, jis įveda naujus duomenis, juos išsaugo
Alternatyvūs scenarijai	- Vartotojo duomenys nekorektiški - Vartotojas atšaukia pakeitimą

1.13 lentelė. Panaudojimo atvejo „Tvarkyti sistemos duomenis“ pradinė specifikacija

Tikslas	Atnaujinti sistemoje esančią informaciją
Aktorius	Administratorius
Ryšiai su kitais panaudojimo atvejais	-
Nefunkciniai reikalavimai	Patogi sąsaja, saugumas
Prieš sąlygos	Administratorius prisijungęs prie sistemos
Sužadinimo sąlyga	Pasirenka atitinkamo komponento atnaujinimo funkciją
Po sąlyga	Pasirinkto komponento duomenys atnaujinti
Pagrindinis scenarijus	Administratorius pasirenka atitinkamą komponentą ir kokius veiksmus su juo atlikti, įveda naują informaciją ir ją patvirtina.
Alternatyvūs scenarijai	- Vartotojas atšaukia atnaujinimą

1.14 lentelė. Panaudojimo atvejo „Įdiegti komponentus“ pradinė specifikacija

Tikslas	Papildyti sistemą naujais komponentais
Aktorius	Administratorius
Ryšiai su kitais panaudojimo atvejais	Tvarkyti sistemos duomenis
Nefunkciniai reikalavimai	Stabilumas, patogumas, patikimumas
Prieš sąlygos	Administratorius prisijungęs prie sistemos
Sužadinimo sąlyga	Pasirenka sistemos komponentų įdiegimą
Po sąlyga	Komponentas įdiegtas
Pagrindinis scenarijus	Vartotojas pasirenka komponentą ir atlieka įdiegimo veiksmą
Alternatyvūs scenarijai	-

1.15 lentelė. Panaudojimo atvejo „Ištrinti komponentus“ pradinė specifikacija

Tikslas	Pašalinti nereikalingus sistemai komponentus.
Aktorius	Administratorius
Ryšiai su kitais panaudojimo atvejais	Tvarkyti sistemos duomenis
Nefunkciniai reikalavimai	Stabilumas, patogumas, patikimumas
Prieš sąlygos	Administratorius prisijungęs prie sistemos
Sužadinimo sąlyga	Pasirenka sistemos komponentų šalinimą.
Po sąlyga	Komponentas pašalintas
Pagrindinis scenarijus	Vartotojas pasirenka komponentą ir atlieka šalinimo veiksmą
Alternatyvūs scenarijai	-

1.16 lentelė. Panaudojimo atvejo „Valdyti komponento informaciją“ pradinė specifikacija

Tikslas	Atnaujinti komponentui priklausančią informaciją.
Aktorius	Administratorius
Ryšiai su kitais panaudojimo atvejais	Tvarkyti sistemos duomenis
Nefunkciniai reikalavimai	Stabilumas, patogumas, patikimumas
Prieš sąlygos	Administratorius prisijungęs prie sistemos
Sužadinimo sąlyga	Pasirenka sistemos komponentų informacijos valdymą.
Po sąlyga	Komponento informacija atnaujinta.
Pagrindinis scenarijus	Vartotojas pasirenka komponentą ir redaguoja jam priklausančią informaciją.
Alternatyvūs scenarijai	-

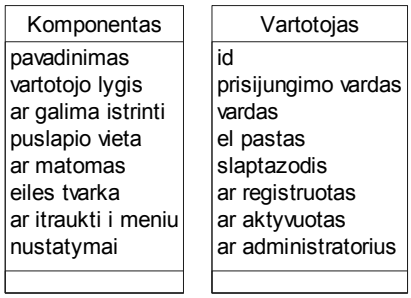
1.17 lentelė. Panaudojimo atvejo “Valdyti komponentą” pradinė specifikacija

Tikslas	Valdyti komponento atvaizdavimą sistemoje.
Aktorius	Administratorius
Ryšiai su kitais panaudojimo atvejais	Tvarkyti sistemos duomenis
Nefunkciniai reikalavimai	Stabilumas, patogumas, patikimumas
Prieš sąlygos	Administratorius prisijungęs prie sistemos
Sužadinimo sąlyga	Pasirenka sistemos komponentų valdymą
Po sąlyga	Komponento atvaizdavimo informacija atnaujinta.
Pagrindinis scenarijus	Vartotojas pasirenka komponentą ir nustato, jo eiliškumą, kurioje tinklapio vietoje jį atvaizduoti, ar komponentas bus matomas puslapyje ir, ar neprijungęs vartotojas jį galės matyti.
Alternatyvūs scenarijai	-

1.10. Reikalavimai duomenims

Norint įgyvendinti komponentinę, duomenų struktūra turi būti griežtai standartizuota tiek bendrai visai sistemai, tiek kiekvienam komponentui atskirai. Sistemai bei atskiriems sistemos komponentams reikalingi duomenys saugomi duomenų bazėje.

Abstrakčią karkaso duomenų struktūrą sudaro tik dvi lentelės. Integravus papildomus komponentus, jų atsiras daugiau priklausomai nuo kiekvieno komponento. Karkaso duomenų struktūra pavaizduota 1.9 paveiksle.



1.9 pav. Abstrakti karkaso duomenų struktūra

Karkasas saugo duomenis apie komponentus, sudarančius patį karkasą ir sukurtą interneto sistemą Taip pat saugo informaciją apie sistemos vartotojus.

Duomenys apie komponentus:

- Komponento pavadinimas;
- Kokias teises turintis vartotojas gali matyti komponentą (paprastas, registruotas ar administratorius);
- Ar komponentą galima ištrinti iš sistemos;
- Jei komponentas rodomas puslapyje, kurioje puslapio pusėje (kairėje ar dešinėje) jis matomas.
- Kokia eilės tvarka rodomas komponentas (kuo mažesnė reikšmė tuo didesnė rodymo pirmenybė).
- Ar komponentą įtraukti į meniu;
- Komponento nustatymai, atsižvelgiant į komponento paskirtį.

Duomenys apie vartotojus:

- Vartotojo identifikacinis numeris sistemoje;
- Vartotojo prisijungimo vardas;
- Vartotojo vardas;
- Elektroninio pašto adresas;
- Ar vartotojas registruotas, ar aktyvuotas, ar turi administratoriaus teises;

1.11. Nefunkciniai reikalavimai ir apribojimai

1.11.1. Reikalavimai standartams

Komponentas privalo turėti tokias savybes:

- Specifikaciją;
- Sąsaja su išore;
- Įgyvendinimą;

Pats komponentas turi priklausyti tik nuo specifikacijos – jei jis priklausys ir nuo įgyvendinimo, tai šio komponento pakeitimo ar pakartotinio panaudojimo galimybės sumažės [4].

Turi būti kompromisas tarp naudojimo ir pakartotinio naudojimo.

Naršyklei pateikiamas turinys turi atitikti W3C rekomendacijas XHTML [9] bei CSS [10] struktūrai.

1.11.2. Reikalavimai veikimui

- Sistema turi dirbti stabiliai;
- Komponentų keitimas, įdiegimas bei šalinimas vartotojui turi būti nepastebimi veikimo atžvilgiu.

1.11.3. Kiti reikalavimai

Sistema turi būti patikima. Labai svarbu, kad sistemą būtų galima lengvai plėsti – tokiu būdu kiekvienas galės pritaikyti sistemą savo poreikiams naudodamas jau sukurtus komponentus bei kurdamas savus. Vartotojų duomenys turi būti apsaugoti, t.y. kiti jų matyti negali, nebent pats vartotojas tai leidžia.

1.12. Rizikos faktorių analizė

Kuriant naujovišką sistemą, visada tikslinga išnagrinėti rizikos faktorius ir numatyti eliminavimo arba sumažinimo būdus. Kuriant komponentinę sistemą, rizika atsiranda dėl šių sistemos savybių:

- Sunku sukurti pakartotinai naudojamą komponentą, kuris atitiktų įvairių sistemų reikalavimus. Tai užima daug laiko.
- Sistemos dydžio kitimas – sistema nuolat papildoma naujais komponentais arba sumažinama jos apimtis ištrinant nereikalingus komponentus;
- Informacinių technologijų naujumas;
- Žinių ir įgūdžių trūkumas;

Rizikos faktorių eliminavimo būdai pateikti 1.18 lentelėje:

1.18 lentelė. Rizikos faktorių analizė

Rizikos faktorius	Rizikos faktoriaus eliminavimo būdas
Nespėsime sukurti visų reikalingų pakartotinai panaudojamų komponentų	Susidarysime sistemos kūrimo grafiką ir jo laikysimės.
Tokią sistemą kuriame pirmą kartą, todėl gali nepavykti jos įgyvendinti.	Išanalizavę, kaip sukurtos panašios sistemos, paimsime visas gerąsias jų savybes. Analizuojame komponentinių sistemų kūrimo metodus ir gilinamės į siūlomus sprendimus iškilusioms problemoms spręsti.
Sistema gali neveikti įdiegus arba išėmus komponentus.	Sukursime komponentinę sistemą, kurioje skirtingi komponentai nepriklausys vienas nuo kito, todėl sistemos darbas nesutriks.
Sistema gali neveikti skirtingose platformose (Windows, Linux)	Naudosime tik abejose platformose palaikomas PHP funkcijas.
Naršyklė gali neteisingai atvaizduoti komponentus.	Kurdami laikysimės XHTML standarto.

1.13. Rezultato kokybės kriterijai

Svarbiausi komponentinių sistemų kūrimo karkaso kokybės kriterijai:

- Sistemos veiksmingumas:
  - Ar sukurta sistema veikia tinkamai?
  - Ar išsprendžia išsikeltas problemas?
- Sistemos našumas:
  - Ar sistema suprojektuota ir realizuota optimaliai, t.y. ar nevyksta pertekliniai procesai?
- Universalumas:
  - Ar sistemą galima visiškai ar bent iš dalies pritaikyti įvairioms interneto sistemoms kurti?
- Išplečiamumas:
  - Ar sistema lengvai plečiama modifikuojant jau sukurtus komponentus arba pridedant naujus komponentus?
  - Kokios galimybė kurti naujus komponentus?
- Patikimumas:
  - Ar panaikinti arba sumažinti rizikos faktoriai?
  - Ar sistema veikia be klaidų?
- Ar bent minimaliai apsaugoti vartotojų asmeniniai duomenys?
  - Naudojimosi paprastumas ir lengvumas:
  - Ar visi komponentai būtinai reikalingi?
  - Ar sistemą sudėtinga pritaikyti vartotojo reikmėms?
- Sistemos efektyvumas:
  - Ar sumažėjo sistemos kūrimo laikas naudojant komponentus?

1.14. Analizės išvados

- Literatūroje rasti du metodai komponentinėms interneto sistemoms kurti: komponentais pagrįstos programinės įrangos inžinerija (CBSE) ir šablonų principais paremta programinės įrangos architektūra (POSA). Išanalizavus, kokias problemas jie sprendžia, nuspręsta, kad tikslinga naudoti juos abu, nes apjungus jų gerąsias savybes jie duoda sinergetinį efektą.
- Išnagrinėtos jau sukurtos panašios sistemos – PHP-Nuke, XOOPS, SMART3 ir PRADO, siekiant išsiaiškinti šių sistemų privalumus ir trūkumus. Pastebėjome, jog didžiausias trūkumas tas, kad senai pradėtos kurti sistemos nesilaiko šiuolaikinių kodo rašymo standartų – tai apsunkina naujų komponentų kūrimą ir esamų integravimą į jau gyvuojančias sistemas. Taip pat nenaudojamas vienas iš literatūroje siūlomų metodų – POSA.
- Suformavome pradinę sistemos architektūrą, remdamiesi MVC modeliu ir išsirinkome pagrindines ir pagalbines įgyvendinimo priemones (PHP5, Apache serveris, MySQL duomenų bazė, Smarty - šablonų vaizdavimo sistema, PEAR).
- Apsibrėžėme pagrindines kompiuterizuojamas sistemos funkcijas:
  - Peržiūrėti turinį;
  - Užsiregistruoti;
  - Prisijungti;
  - Tvarkyti savo informaciją;
  - Tvarkyti vartotojų informaciją;
  - Įdiegti komponentą;
  - Ištrinti komponentą;
  - Tvarkyti komponento informaciją;
  - Tvarkyti komponentus.

  Pateikėme pradines panaudojimo atvejų specifikacijas ir nustatėme reikalavimus sistemai, veikimui ir duomenims.
- Išanalizavome galimus rizikos faktorius ir pateikėme sprendimus jiems eliminuoti arba sumažinti.

- Nustatėme rezultato kokybės faktorius, kurie padės įvertinti karkaso naudingumą. Pagrindiniai kokybės faktoriai:
  - Sistemos veiksmingumas;
  - Sistemos našumas;
  - Universalumas;
  - Išplečiamumas;
  - Patikimumas;
  - Naudojimosi paprastumas ir lengvumas;
  - Sistemos efektyvumas.

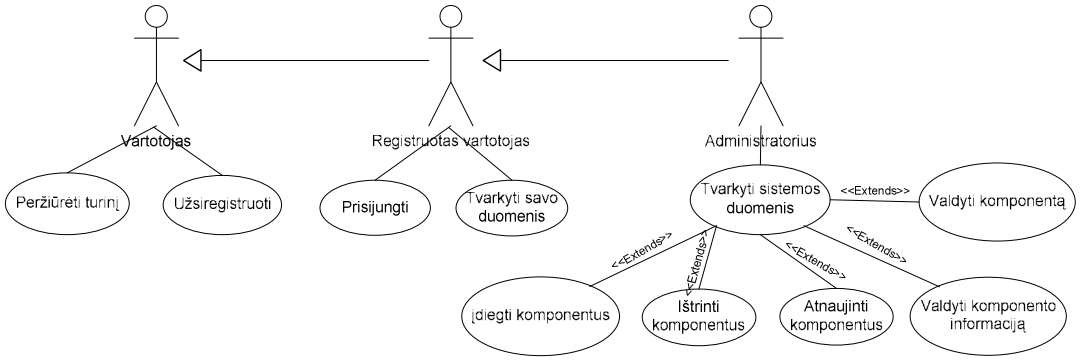
2. KOMPONENTINIŲ INTERNETO SISTEMŲ KŪRIMO KARKASO REIKALAVIMAI

2.1. Reikalavimų specifikacija

Sistemoje galima išskirti 3 tipų aktorius:

- Paprastas vartotojas;
- Registruotas vartotojas;
- Sistemos administratorius;

Sistemos panaudojimo atvejai sistemos atžvilgiu pateikti 2.1 paveiksle.

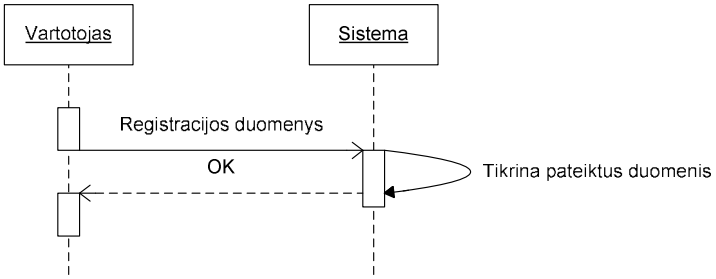


2.1 pav. Sistemos panaudojimo atvejai

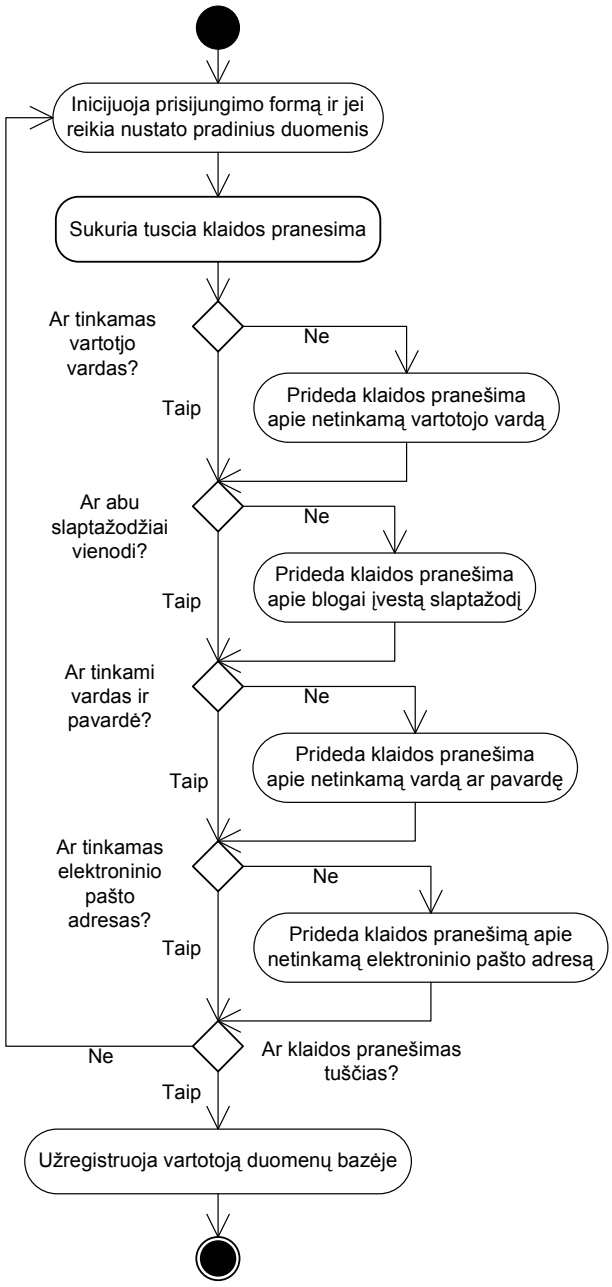
Šių sistemos panaudojimo atvejų specifikacijos pateiktos 2.1 – 2.8 lentelėse.

2.1 lentelė Panaudojimo atvejo „Užsiregistruoti“ specifikacija

Prieš sąlyga	Vartotojas neregistruotas.
Įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas inicijuoja registracijos formą.	1.1 Sistema parodo registracijos formą.
2. Vartotojas įveda vartotojo vardą, slaptažodį, pakartoja slaptažodį, taip pat savo vardą, pavardę, elektroninio pašto adresą ir paspaudžia užsiregistravimo mygtuką.	2.1 Sistema tikrina ar tokio vartotojo dar nėra. Jei toks vartotojas yra sistema parodo klaidos pranešimą ir grįžta į 1.1 žingsnį. 2.2 Sistema tikrina ar vartotojo vardą sudaro simboliai iš leistinų simbolių aibės. Jei vartotojo varde yra simbolių iš neleistinų simbolių aibės, parodomas klaidos pranešimas ir sistema grįžta į 1.1 žingsnį. 2.3 Sistema tikrina ar sutampa įvesti slaptažodžiai. Jei slaptažodžiai nesutampa, sistema parodo klaidos pranešimą ir grįžta į 1.1 žingsnį. 2.4 Sistema tikrina ar vardą ir pavardę sudaro simboliai iš leistinų simbolių aibės Jei vardas ar pavardė turi simbolių iš neleistinų simbolių aibės sistema parodo klaidos pranešimą ir grįžta į 1.1 žingsnį. 2.5 Sistema tikrina ar elektroninio pašto adresą sudaro leistinų simbolių aibė, patikrina ar elektroninio pašto adresas atitinka reikalaujamą standartą. Jei elektroninio pašto adresas turi simbolių iš neleistinų simbolių aibės ar jo formatas neatitinka standarto, parodo klaidos pranešimas ir grįžta į 1.1 žingsnį.
3. Vartotojas patvirtina duomenis	3.1 Sistema baigia darbą.
Po sąlyga	Vartotojas tampa registruotu vartotoju



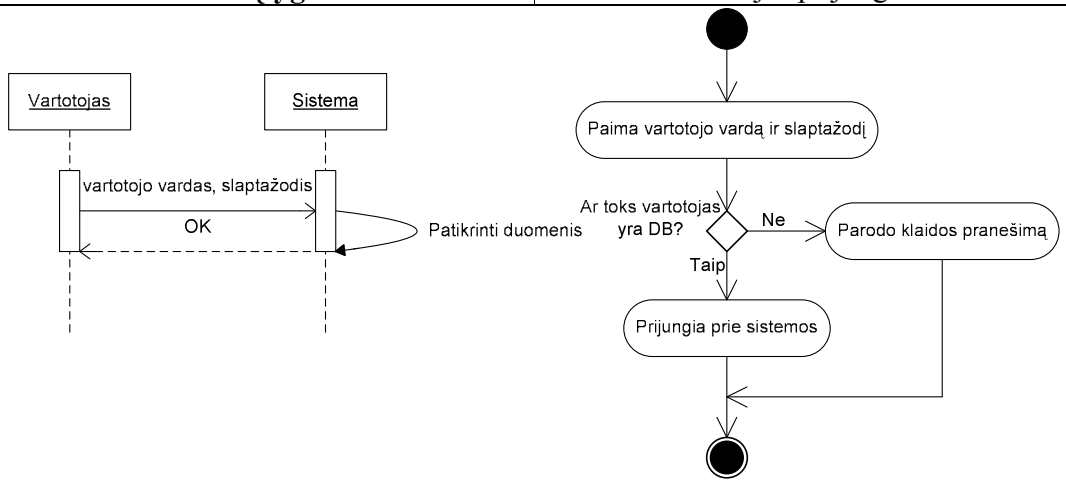
2.2 pav Panaudojimo atvejo „Užsiregistruoti“ sekų diagrama



2.3 pav Panaudojimo atvejo „Užsiregistruoti“ veiklos diagrama

2.2 lentelė Panaudojimo atvejo „Prisijungti“ specifikacija

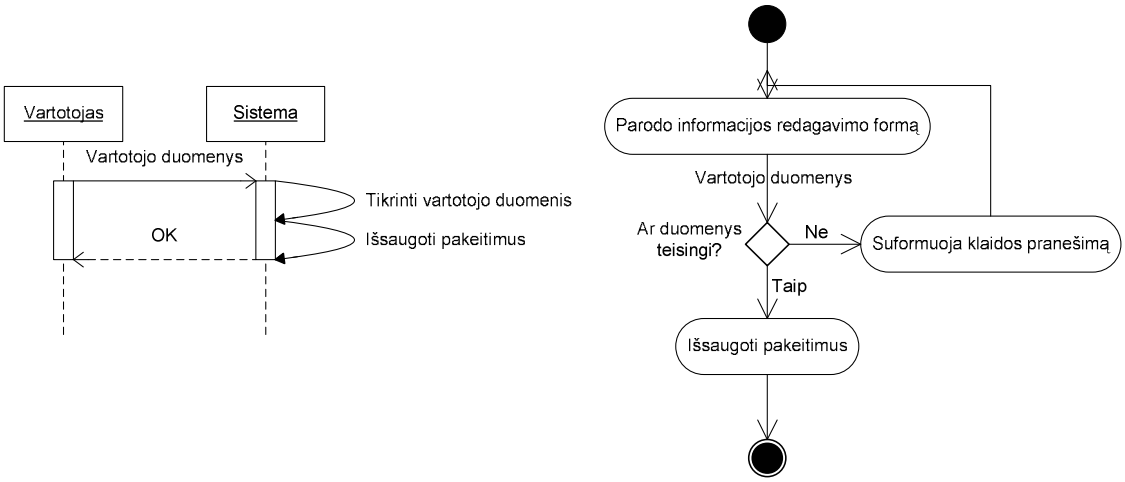
Prieš sąlyga	Vartotojas užsiregistravęs sistemoje
Įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas įveda vartotojo vardą ir slaptažodį	1.1 Sistema patikrina ar duomenys teisingi Jei duomenys neteisingi, sistema parodo klaidos pranešimą ir baigia darbą.
Po sąlyga	Vartotojas prijungiamas



2.4 pav. Panaudojimo atvejo „Prisijungti“ sekų (kairėje) ir veiklos (dešinėje) diagramos

2.3 lentelė Panaudojimo atvejo „Tvarkyti savo duomenis“ specifikacija

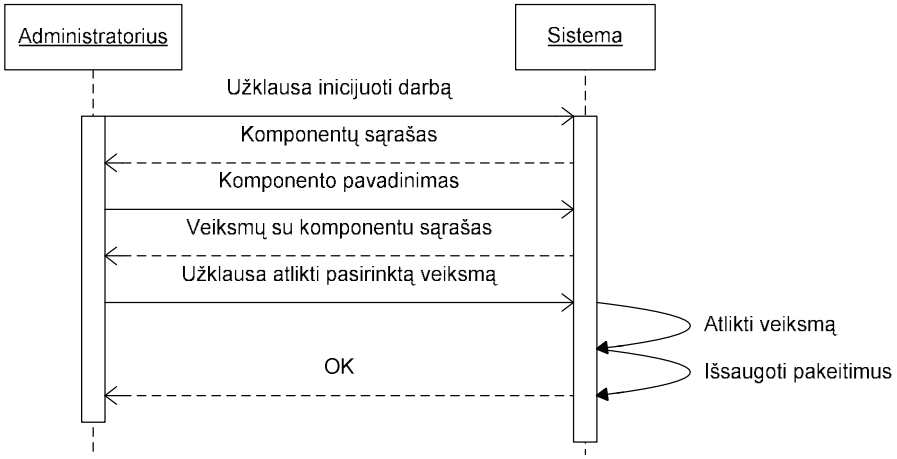
Prieš sąlyga	Vartotojas prisijungęs prie sistemos
Įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas inicijuoja duomenų apie save redagavimo formą.	1.1 Sistema atidaro duomenų vartotoją redagavimo formą.
2. Vartotojas pakeičia norimus duomenis	
3. Vartotojas patvirtina naujus duomenis	3.1 Sistema patikrina duomenis ir juos išsaugo.
Po sąlyga	Vartotojo duomenys atnaujinti



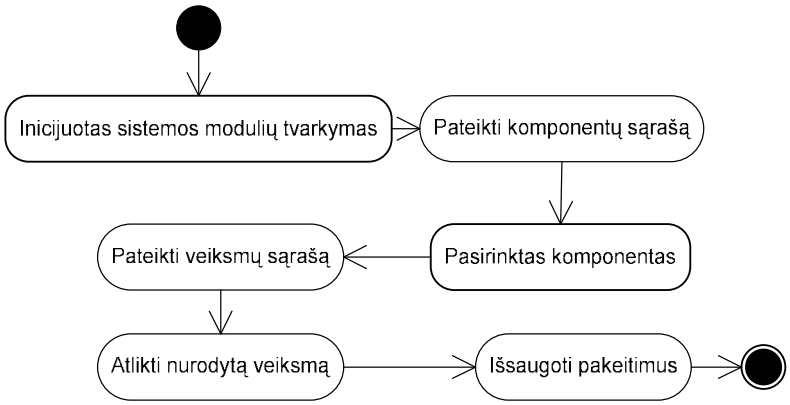
2.5 pav. Panaudojimo atvejo „Tvarkyti savo duomenis“ sekų (kairėje) ir veiklos (dešinėje) diagramos

2.4 lentelė Panaudojimo atvejo „Tvarkyti sistemos duomenis“ specifikacija

Prieš sąlyga	Administratorius prisijungęs prie sistemos.
Įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Administratorius pasirenka norimą modulį	1.1 Sistema pateikia galimų veiksmų sąrašą.
2. Administratorius pasirenka atitinkamą veiksmą	2.2 Sistema atlieka pasirinktą veiksmą. 2.3 Sistema išsaugo duomenis 2.4 Sistema baigia darbą
	2.1 Sistema išsaugo duomenis 2.1 Sistema baigia darbą
Po sąlyga	Pasirinkto modulio duomenys atnaujinti



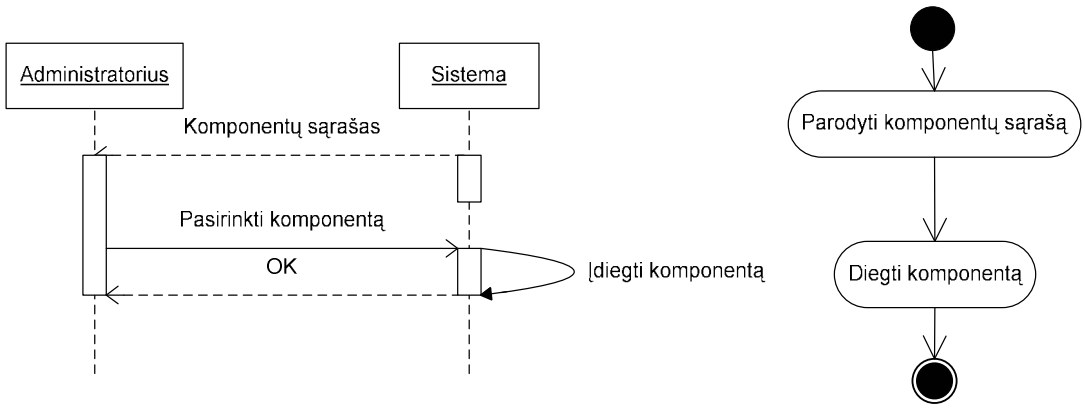
2.6 pav. Panaudojimo atvejo „Tvarkyti sistemos duomenis“ sekų diagrama



2.7 pav. atvejo „Tvarkyti sistemos duomenis“ veiklos diagrama

2.5 lentelė Panaudojimo atvejo „Įdiegti komponentą“ specifikacija

Prieš sąlyga	Administratorius prisijungęs prie sistemos;
Įvykių srautas	**Sistemos reakcija ir sprendimai**
	1. Sistema parodo komponentų sąrašą
2. Administratorius pasirenka, kurį komponentą įdiegti	2.1 Sistema įdiegia komponentą 2.2 Sistema baigia darbą.
Po sąlyga	Komponentas įdiegtas



2.8 pav. Panaudojimo atvejo „Įdiegti komponentą“ sekų (kairėje) ir veiklos (dešinėje) diagramos

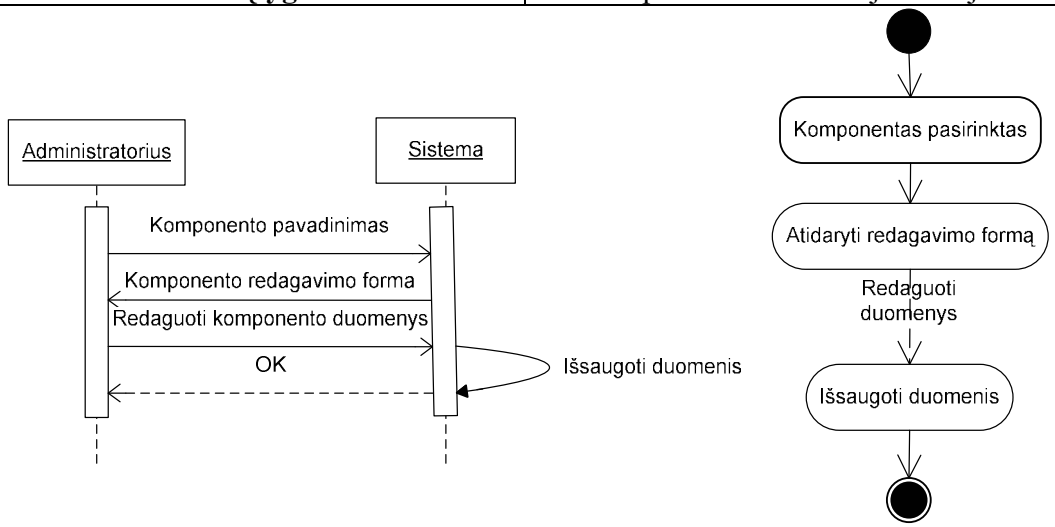
2.6 lentelė Panaudojimo atvejo „Ištrinti komponentą“ specifikacija

Prieš sąlyga	Administratorius prisijungęs prie sistemos.
Įvykių srautas	**Sistemos reakcija ir sprendimai**
	1. Parodyti komponentų sąrašą
2. Administratorius pasirenka, kurį komponentą ištrinti	2.1 Sistema ištrina komponentą 2.2 Sistema baigia darbą.
Po sąlyga	Komponentas ištrintas

Panaudojimo atvejo „Ištrinti komponentą“ sekų ir veiklos diagramos analogiškos panaudojimo atvejo „Įdiegti komponentą“ sekų ir veiklos diagramoms pavaizduotoms 2.8 paveiksle.

2.7 lentelė Panaudojimo atvejo „Valdyti komponento informaciją“ specifikacija

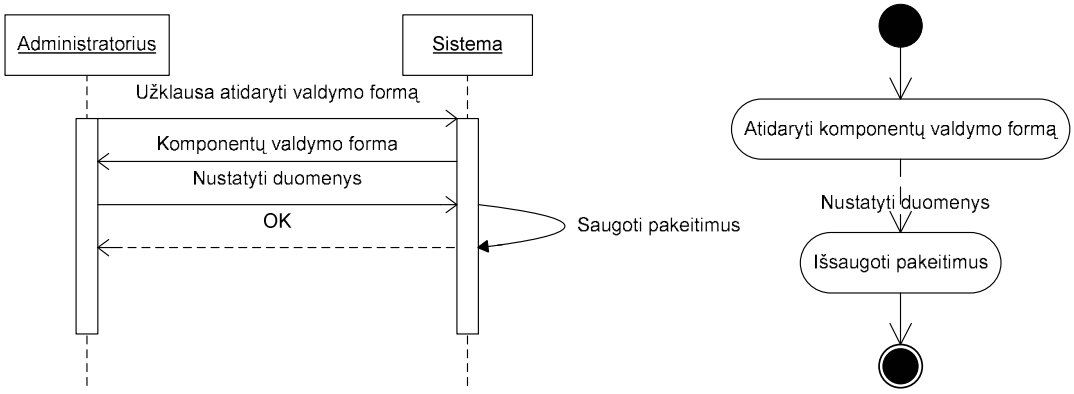
Prieš sąlyga	Administratorius prisijungęs prie sistemos.
Įvykių srautas	**Sistemos reakcija ir sprendimai**
	1. Sistema pateikia komponentų sąrašą
2. Administratorius pasirenka, kurio komponento informaciją redaguoti	2.1 Sistema atidaro redagavimo formą
3. Administratorius redaguoja komponento informaciją ir patvirtina savo veiksmus	3.1 Sistema išsaugo duomenis 3.2 Sistema baigia darbą
Po sąlyga	Komponento informacija atnaujinta



2.9 pav. Panaudojimo atvejo „Valdyti komponento informaciją“ sekų (kairėje) ir veiklos (dešinėje) diagramos

2.8 lentelė Panaudojimo atvejo „Valdyti komponentą“ specifikacija

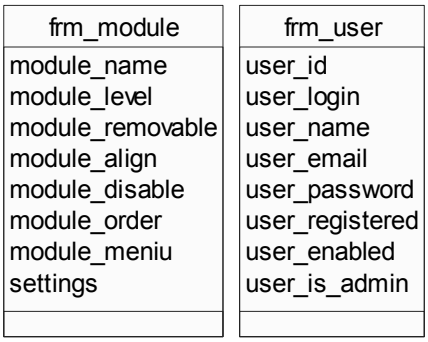
Prieš sąlyga	Administratorius prisijungęs prie sistemos.
Įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Administratorius inicijuoja komponentų valdymo formą	1.1 Sistema atidaro komponentų valdymo formą.
2. Administratorius parenka atitinkamo komponento rodymo vietą 3. Administratorius nustato komponento eiliškumą. 4. Administratorius nustato ar komponentas bus matomas puslapyje ar ne. 5. Administratorius nustato, kuriuos komponentus gali matyti neprisijungęs vartotojas.	
6. Administratorius patvirtina savo veiksmus	6.1 Sistema išsaugo duomenis 6.2 Sistema baigia darbą
Po sąlyga	Komponento informacija atnaujinta



2.10 pav Panaudojimo atvejo „Valdyti komponentą“ sekų (kairėje) ir veiklos (dešinėje) diagramos

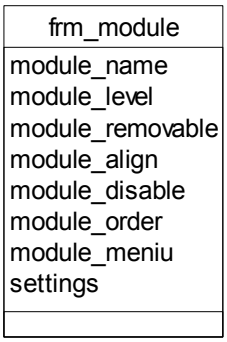
2.2. Karkaso duomenų struktūros

Karkaso duomenų struktūrą sudaro tik 2 lentelės – frm_user ir frm_module. Visos kitos lentelės atsiranda instaliavus papildomus komponentus. PHP komponentą sudarančios lentelės pavaizduotos 2.11 paveiksle.



2.11 pav. Karkaso duomenų struktūra

2.2.1. Lentelė frm_module (2.12 pav.)



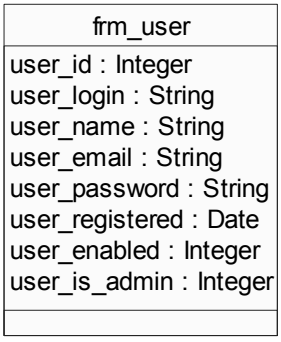
2.12 pav. Lentelė frm_module

Šioje lentelėje saugomi sistemoje dalyvaujančių komponentų sąrašas ir jų nustatymai. Lentelės atributai aprašyti 2.9 lentelėje.

2.9 lentelė. Lentelės frm_module atributai.

Atributas	Atributo paskirtis
module_name	Komponento pavadinimas
module_level	Nurodyti, kokie sistemos vartotojai matys komponentą. Jei atributo reikšmė lygi 0 – komponentą galės matyti visi sistemos vartotojai, 1 – tik užsiregistravę vartotojai, 2 – tik sistemos administratorius
module_removable	Nusakyti, ar komponentą galima pašalinti iš sistemos. Jei atributo reikšmė lygi 1 – komponentą pašalinti galima, jei 0 – negalima.
module_align	Nusakyti, kurioje puslapio pusėje (dešinėje ar kairėje) bus atvaizduotas komponentas.
module_disabled	Nustatyti, ar modulis bus rodomas puslapyje. Jei atributo reikšmė lygi 1 – komponentas matomas puslapyje, jei 0 – ne.
module_order	Nurodyti modulio eiliškumą (1-999). Mažesnis skaičius reiškia pirmenybę.
module_meniu	Nurodyti, ar komponentas bus matomas kaip meniu punktas.
module_settings	Komponento nustatymai.

2.2.2. Lentelė frm_user (2.13 pav.)



2.13 pav. Lentelė frm_user

Šioje lentelėje saugoma registruotų sistemos vartotojų registracija. Lentelės atributai aprašyti 2.10 lentelėje.

2.10 lentelė. Lentelės frm_user atributai

Atributas	Atributo paskirtis
user_id	vartotojo id
user_login	vartotojo prisijungimo vardas
user_name	vartotojo pilnas vardas
user_email	vartotojo elektroninis paštas
user_password	vartotojo prisijungimo slaptažodis
user_registered	vartotojo prisiregistravimo data
user_enabled	ar vartotojas aktyvuotas? Jei atributo lygi 1 – vartotojas gali prisijungti prie sistemos, jei 0 – negali.
user_is_admin	ar vartotojas yra administratorius

3. PHP KOMPONENTŲ KARKASO PROJEKTAS

3.1. PHP komponentų karkaso pagrindimas ir esmės išdėstymas

Kaip jau minėjome ankstesniuose skyriuose, interneto sistemų kūrimas apima analizę, projektavimą, dizaino kūrimą, realizavimą ir patalpinimą serveryje. Svarbiausia iš šių dalių yra realizavimas, kadangi šiandien tai daugiausiai laiko ir pinigų atimanti dalis. Sparčiai didėjant rinkos evoliucijai pastebimas greitas programinių produktų sudėtingumo ir apimčių augimas. Dėl šios priežasties skirtingoms sistemoms kurti pritaikomi tie patys veikimo principai, tačiau jos kuriamos iš naujo, o identiškos savybės, priklausomai nuo sistemos, elgiasi skirtingai. Taip pat tokių sistemų palaikymas reikalauja daug pastangų ir išteklių.

Atsižvelgiant į šias problemas, atsiranda poreikis produktui, kurio pagalba būtų galima greitai ir efektyviai kurti interneto sistemas, kurios laikytųsi šiuolaikinių standartų, būtų lengvai keičiamos ir jų palaikymas nereikalautų daug pastangų ir išteklių.

Dėl šios priežasties, bus suprojektuotas ir sukurtas komponentinių interneto sistemų kūrimo karkasas, vadovaujantis pakartotinio naudojimo ir komponentų principais. Į jį bus įtraukti pradiniai komponentai, reikalingi interneto sistemoms kurti. Karkasas bus lengvai plečiamas pridedant papildomus komponentus. Komponentų funkcionalumas nepriklausys vienas nuo kito. Jų visuma sudarys vieningą veikiančią sistemą, tačiau ištrynus bent vieną jų, sistemos darbas nebus sutrikdytas. Vartotojo sąsaja, veikimo mechanizmas ir veiksmų kontrolė bus griežtai atskirti vieni nuo kitų, todėl tolimesnis sukurtos sistemos palaikymas nereikalaus didelių pastangų.

3.2. Sistemos architektūra - statinės struktūros modelis

Sistemą sudaro 4 lygmenys:

- Vartotojo sąsaja;
- Valdymo lygmuo (Kontroleris);
- Logikos lygmuo (Modelis).
- Duomenų lygmuo (Duomenų bazė);

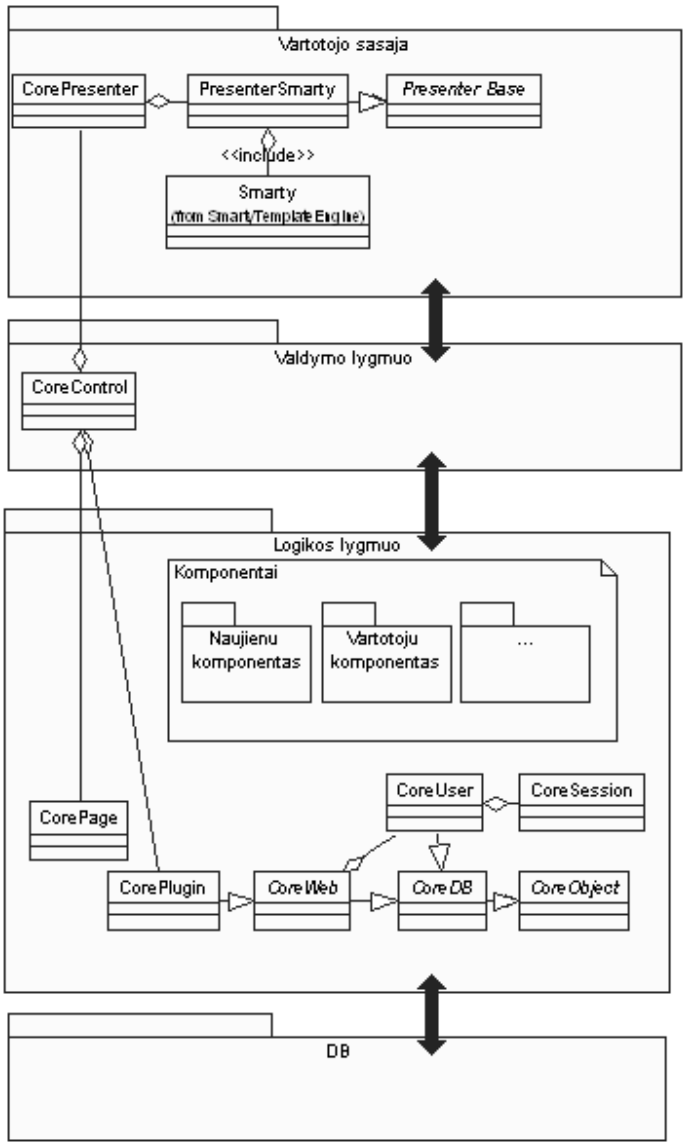
Vartotojo sąsaja apima informacijos atvaizdavimą bei pateikia vartotojo duomenų įvedimo formas.

Valdymo lygmuo – tai tarpinis lygmuo tarp vartotojo sąsajos ir logikos lygmens. Inicijuoja atitinkamus logikos lygmens komponentus, priklausomai nuo vartotojo užklausos.

Logikos lygmuo – tai visi sistemoje atliekami veiksmai su duomenimis, gautais iš duomenų lygio, vartotojo užklausų apdorojimas bei apdorotos informacijos pateikimas vartotojo sąsajos lygmeniui.

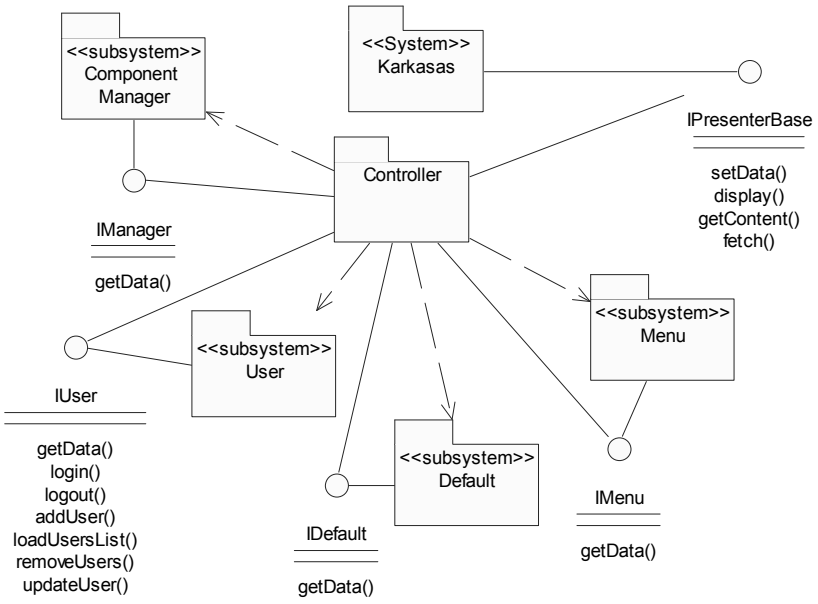
Duomenų lygmenyje atliekami veiksmai su duomenų baze (informacijos išrinkimas, redagavimas, šalinimas ir pan.).

3.2.1. Loginė PHP komponentų karkaso architektūra



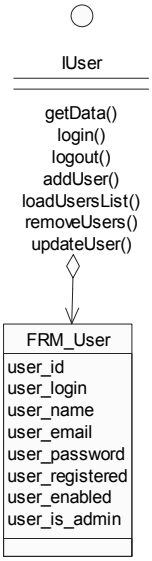
.1 pav. Loginė PHP komponentų karkaso architektūra

.1 paveiksle pavaizduota loginė karkaso architektūra. Karkasą sudaro vartotojo sąsajos (View), valdymo (Controller), logikos (Model) ir duomenų bazės lygmenys. Šios dalys tarpusavyje bendrauja tiesiogiai ir per sąsajas. Grafiškai tai pavaizduota 3.2 paveiksle.

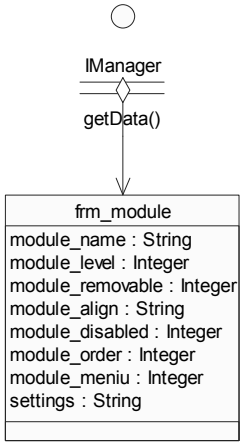


3.2 pav. PHP komponentų karkaso sąsajos

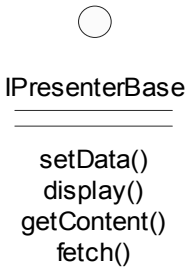
Plačiau įsigilinus į sąsajas, galima atvaizduoti kiekvienos sąsajos informacinius modelius.



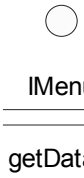
3.3 pav. Vartotojų valdymo interfeiso informacinis modelis



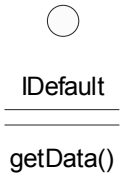
3.4 pav. Komponentų valdymo interfeiso informacinis modelis.



3.5 pav. Karkaso interfeiso IpresenterBase interfeiso informacinis modelis



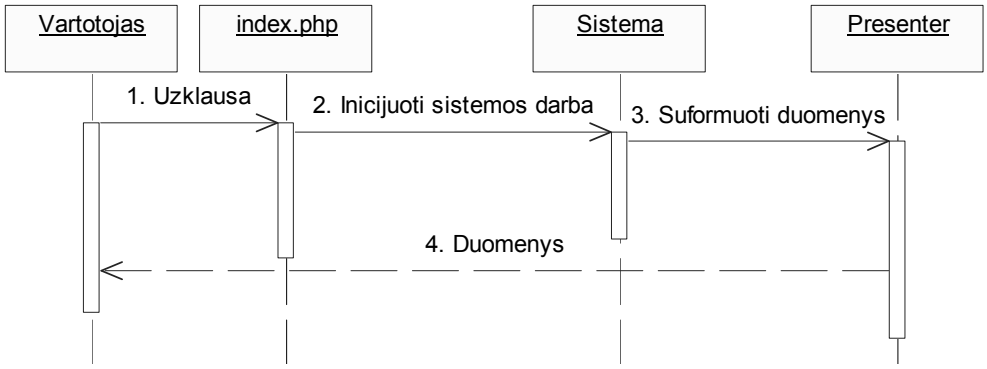
3.6 pav. Meniu valdymo interfeiso informacinis modelis



3.7 pav. Papildomo komponento interfeiso informacinis modelis

3.2.2. Vartotojo sąsajos lygmuo

Vartotojo sąsajos (View) lygmuo yra skirtas interaktyviam bendravimui su vartotoju. Jame pagal vartotojo užklausą atvaizduojama iš sistemos modulių gauta informacija HTML arba XML formatu. Grafiškai tai pavaizduota 3.8 paveiksle, nesigilinant į kitų lygmenų darbą.



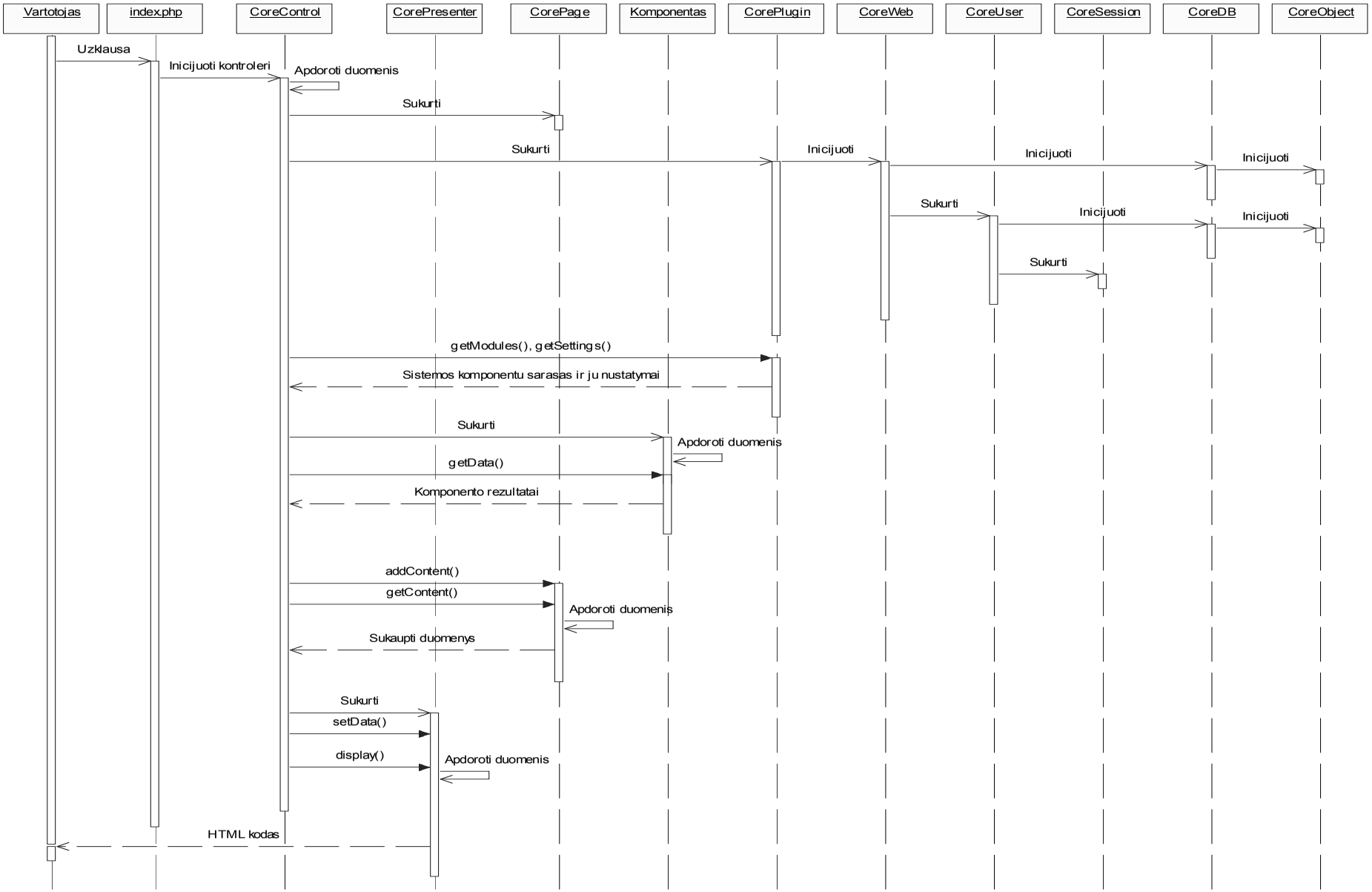
3.8 pav. Vartotojo sąsajos (View) lygmens sekų diagrama

3.2.3. Valdymo ir logikos lygmenys

Valdymo (Controller) lygmuo atlieka pradinį vartotojo siunčiamų duomenų apdorojimą, inicializuoja už vartotojo sąsajos pateikimą atsakingą objektą ir pagal vartotojo užklausą reikalingus sistemos komponentus informacijai pateikti. Gautą informaciją perduoda vartotojo sąsajos objektui ir inicijuoja informacijos atvaizdavimo metodą.

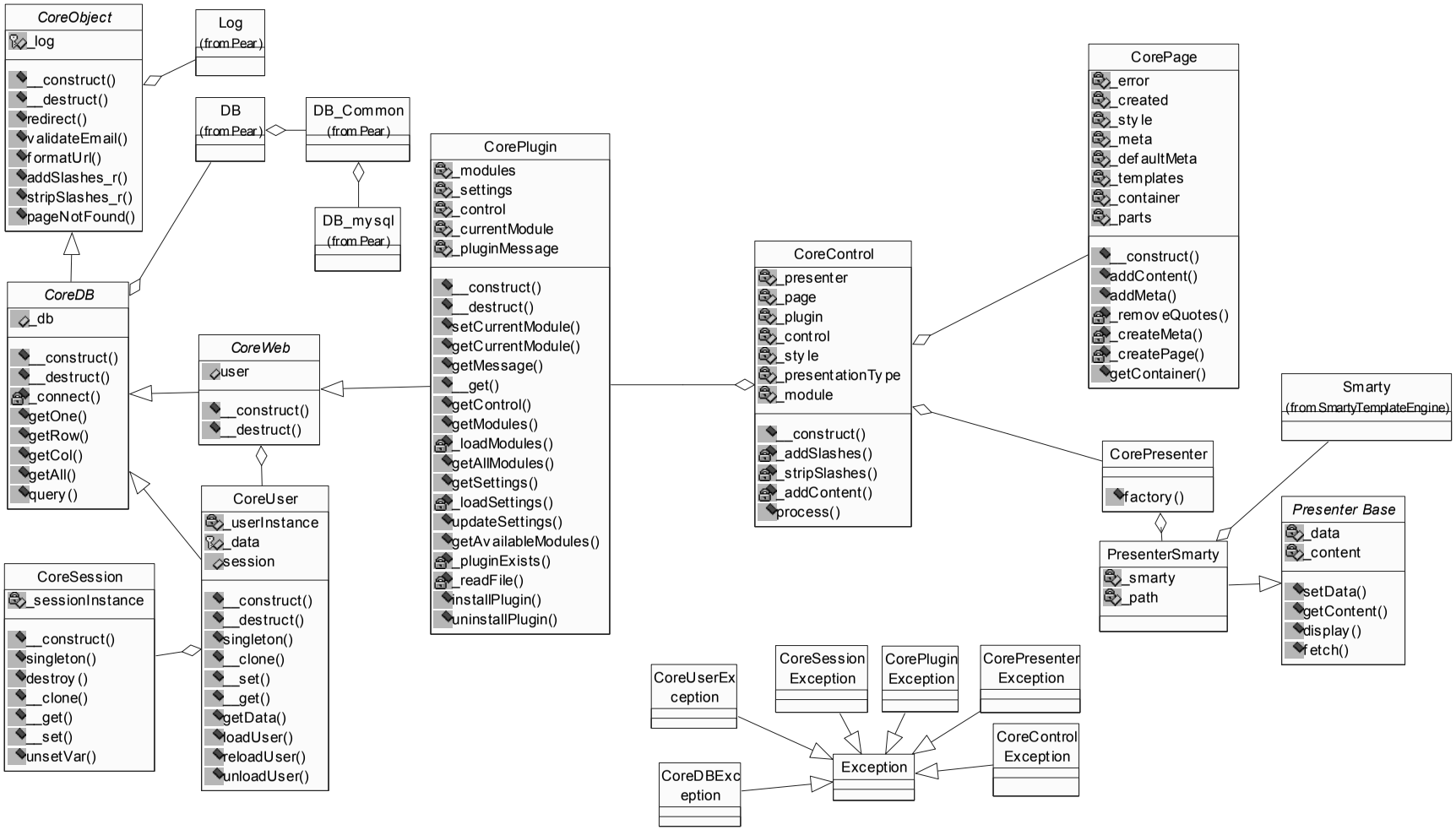
Logikos (Model) lygmuo atsakingas už informacijos pateikimą pagal vartotojo užklausą. Jį sudaro sistemos branduolys ir prie jo prijungti sistemos komponentai, kurie atlieka specifinius veiksmus informacijai gauti. Sistemos branduolys turi klasę, skirtą manipuliuoti duomenų bazėje esančia informacija.

Grafiškai vaizduojant, abu lygmenys yra neatsiejami vienas nuo kito, todėl jie pavaizduoti kartu 3.9 paveiksle.



3.9 pav. Valdymo ir logikos lygmenų sekų diagrama

3.3. Detalus PHP komponentų karkaso projektas



3.10 pav. Detali PHP komponentų karkaso klasių diagrama

3.3.1. Klasė CoreObject

3.1 lentelė Klasės CoreObject aprašymas

Aprašymas
Abstrakti pamatinė sistemos klasė. Turi bendrus metodus, kuriais gali naudotis visi komponentai
Ryšiai su kitomis klasėmis
Agregavimo ryšys su PEAR klase *Log*
Atributai
protected $_log – PEAR klasės *Log* objektas

3.2 lentelė Klasės CoreObject metodas __construct()

public __construct() - klasės konstruktorius
Santrauka
void __construct ()
Aprašymas
Klasės konstruktorius. Atributui *$_log* priskiria PEAR klasės *Log* objektą.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.3 lentelė Klasės CoreObject metodas __destruct()

public __destruct() - klasės destruktorius
Santrauka
void __destruct ()
Aprašymas
Klasės destruktorius. Sunaikina atributą *$_log*.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.4 lentelė Klasės CoreObject metodas redirect()

public redirect() – nukreipia puslapį pagal nuorodą
Santrauka
void redirect (string $link)
Aprašymas
Naršyklės *header* daliai nusiunčia komandą nukreipiančia puslapį į *$link* nuorodą.
Parametrai
string $link – nuoroda
Grąžinama reikšmė
Nėra

3.5 lentelė Klasės CoreObject metodas validateEmail()

public validateEmail() – tikrina elektroninio pašto adresą
Santrauka
mixed validateEmail (string $email)
Aprašymas
Tikrina, ar elektroninio pašto adresas *$email* yra validus.
Parametrai
string $email – elektroninio pašto adresas
Grąžinama reikšmė
True, jei elektroninio pašto adresas *$email* validus Klaidos pranešimo kodą kitu atveju

3.6 lentelė Klasės CoreObject metodas formatUrl()

public formatUrl() – bando suformuoti teisingą URL adresą
Santrauka
string formatUrl (string $url)
Aprašymas
Išnagrinėja duotą URL adresą *$url* ir jei reikia prideda trūkstamas dalis (pvz: http://).
Parametrai
string $url – URL adresas
Grąžinama reikšmė
Suformuotas URL adresas arba tuščia eilutė, jei adresas buvo blogas

3.7 lentelė Klasės CoreObject metodas addSlashes_r()

public addSlashes_r() – truputi pakeista PHP funkcija addslashes()
Santrauka
mixed addSlashes_r (mixed $data)
Aprašymas
Jei *$data* masyvas – visiems jo elementams pritaiko PHP funkciją *addslashes()*. Jei *$data* eilutė – pritaiko jai PHP funkciją *addslashes()*.
Parametrai
mixed $data – duomenys, kuriuos reikia pakeisti
Grąžinama reikšmė
Masyvas, jei *$data* buvo masyvas Eilutė, jei *$data* buvo eilutė

3.8 lentelė Klasės CoreObject metodas stripSlashes_r()

public stripSlashes_r() – truputi pakeista PHP funkcija stripslashes()
Santrauka
mixed stripSlashes_r (mixed $data)
Aprašymas
Jei *$data* masyvas – visiems jo elementams pritaiko PHP funkciją *stripslashes()*.Jei *$data* eilutė – pritaiko jai PHP funkciją strip*slashes()*
Parametrai
mixed $data – duomenys, kuriuos reikia pakeisti
Grąžinama reikšmė
Masyvas, jei *$data* buvo masyvas Eilutė, jei *$data* buvo eilutė

3.9 lentelė Klasės CoreObject metodas pageNotFound()

public pageNotFound() – grąžina serverio 404 klaidos puslapio duomenų masyvą
Santrauka
array pageNotFound ([string $msg = null])
Aprašymas
Gražina masyvą skirtą 404 klaidos pranešimui parodyti.
Parametrai
string $msg – Klaidos pranešimas
Grąžinama reikšmė
Masyvas su duomenimis, reikalingais klaidos pranešimui parodyti

3.3.2. Klasė CoreDBException

3.10 lentelė Klasės CoreDBException aprašymas

Aprašymas
CoreDB klasės klaidų valdymo klasė
Ryšiai su kitomis klasėmis
Apibendrinimo ryšys su PHP klase *Exception*
Atributai
Nėra

3.3.3. Klasė CoreDB

3.11 lentelė Klasės CoreDB aprašymas

Aprašymas
Abstrakti duomenų bazės valdymo klasė
Ryšiai su kitomis klasėmis
Apibendrinimo ryšys sy klase *CoreObject* ir agregavimo ryšys su PEAR klase *DB*
Atributai
public $_db – PEAR klasės DB_mysql objektas

3.12 lentelė Klasės CoreDB metodas __construct()

public __construct() – klasės CoreDB konstruktorius
Santrauka
void __construct()
Aprašymas
Klasės *CoreDB* konstruktorius. Inicijuoja tėvinės klasės *__construct()* metodą ir klasės metodą *_connect()*.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.13lentelė Klasės CoreDB metodas __destruct()

public __destruct() – klasės CoreDB destruktorius
Santrauka
void __destruct()
Aprašymas
Klasės *CoreDB* destruktorius. Inicijuoja tėvinės klasės *__destruct()* metodą, sunaikina atributą *$_db*.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.14 lentelė Klasės CoreDB metodas _connect()

private _connect() – prijungia prie duomenų bazės
Santrauka
void _connect ()
Aprašymas
Atributui *$_db* priskiria PEAR *DB_mysql* klasės objektą.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.15 lentelė Klasės CoreDB metodas getOne()

public getOne() – grąžina rezultatą iš SELECT užklausos
Santrauka
mixed getOne (string $sq [, mixed $array = array()])
Aprašymas
Grąžina SELECT užklausos pirmos eilutės, pirmo stulpelio reikšmę arba sukuria naują *Exception CoreDBException* objektą jei užklausa turėjo klaidų.
Parametrai
string $sq – SELECT užklausa mixed $array – masyvas, simbolių eilutė arba skaičius, kurie bus pridėti į iš anksto paruoštą užklausą, jei tokia naudojama
Grąžinama reikšmė
SELECT užklausos pirmos eilutės ir pirmo stulpelio reikšmė

3.16 lentelė Klasės CoreDB metodas getRow()

public getRow() – grąžina rezultatą iš SELECT užklausos
Santrauka
array getRow (string $sq [, mixed $array = array()])
Aprašymas
Grąžina masyvą su SELECT užklausos pirmos eilutės reikšmėmis arba sukuria naują *Exception CoreDBException* objektą jei užklausa turėjo klaidų.
Parametrai
string $sq – SELECT užklausa mixed $array – masyvas, simbolių eilutė arba skaičius, kurie bus pridėti į iš anksto paruoštą užklausą, jei tokia naudojama
Grąžinama reikšmė
Masyvas su SELECT užklausos pirmos eilutės reikšmėmis

3.17 lentelė Klasės CoreDB metodas getCol()

public getCol() – grąžina rezultatą iš SELECT užklausos
Santrauka
array getCol (string $sq [, mixed $col = 0 [, mixed $params = array()]])
Aprašymas
Grąžina masyvą su SELECT užklausos pirmo stulpelio reikšmėmis arba sukuria naują *Exception CoreDBException* objektą jei užklausa turėjo klaidų.
Parametrai
string $sq – SELECT užklausa mixed $col – pageidaujamo grąžint stulpelio pavadinimas arba numeris; mixed $params – masyvas, simbolių eilutė arba skaičius, kurie bus pridėti į iš anksto paruoštą užklausą, jei tokia naudojama
Grąžinama reikšmė
Masyvas su SELECT užklausos pirmo stulpelio reikšmėmis

3.18 lentelėKlasės CoreDB metodas getAll()

public getAll() – grąžina rezultatą iš SELECT užklausos
Santrauka
array getAll (string $sq [, mixed $array = array()])
Aprašymas
Grąžina masyvą su SELECT užklausos rezultatu arba sukuria naują *Exception CoreDBException* objektą jei užklausa turėjo klaidų.
Parametrai
string $sq – SELECT užklausa mixed $array – masyvas, simbolių eilutė arba skaičius, kurie bus pridėti į iš anksto paruoštą užklausą, jei tokia naudojama
Grąžinama reikšmė
Masyvas su SELECT užklausos rezultatu

3.19 lentelė Klasės CoreDB metodas query()

public query() – vykdo užklausą duomenų bazėje
Santrauka
mixed query (string $sq [, mixed $array = array()])
Aprašymas
Vykdo užklausą duomenų bazėje. Jei užklausa turėjo klaidų sukuria naują *Exception CoreDBException* objektą.
Parametrai
string $sq – užklausa mixed $array – masyvas, simbolių eilutė arba skaičius, kurie bus pridėti į iš anksto paruoštą užklausą, jei tokia naudojama
Grąžinama reikšmė
Naujas PEAR DB_result objektas - užklausoms, grąžinančioms rezultatą (pvz.: SELECT) PEAR DB_OK – užklausoms, kurios manipuliuoja duomenimis (pvz.: INSERT, DELETE, UPDATE)

3.3.4. Klasė CoreWeb

3.20 lentelė Klasės CoreWeb aprašymas

Aprašymas
Vartotojo informacijos ir sesijos valdymo lygmens klasė
Ryšiai su kitomis klasėmis
Apibendrinimo ryšys su klase *CoreDB* ir agregavimo ryšys su klase *CoreUser*
Atributai
public $_user – klasės *CoreUser* objektas

3.21 lentelė Klasės CoreWeb metodas __construct()

public __construct() – klasės konstruktorius
Santrauka
void __construct ()
Aprašymas
Klasės konstruktorius. Inicijuoja tėvinės klasės *__construct()* metodą, atributui *$_user* priskiria klasės *CoreUser* objektą.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.22 lentelė Klasės CoreWeb metodas __destruct()

public __destruct() – klasės destruktorius
Santrauka
void __destruct ()
Aprašymas
Klasės destruktorius. Inicijuoja tėvinės klasės *__destruct()* metodą.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.3.5. Klasė CoreUserException

3.23 lentelė Klasės CoreUserException aprašymas

Aprašymas
CoreUser klasės klaidų valdymo klasė
Ryšiai su kitomis klasėmis
Apibendrinimo ryšys su PHP klase *Exception*
Atributai
Nėra

3.3.6. Klasė CoreUser

3.24 lentelė Klasės CoreUser aprašymas

Aprašymas
Vartotojo informacijos valdymo klasė
Ryšiai su kitomis klasėmis
Agregavimo ryšys su klase *CoreSession*
Atributai
private $_userInstance – statinis pačios klasės objektas protected $_data – vartotojo duomenų masyvas public $session – klasės *CoreSession* objektas

3.25 lentelė Klasės CoreUser metodas __construct()

public __construct() – klasės konstruktorius
Santrauka
void __construct ()
Aprašymas
Klasės konstruktorius. Inicijuoja tėvinės klasės *__construct()* metodą, atributui *$session* priskiria klasės CoreSession objektą. Jei objektas *$session* turi atributą $user_id arba abu atributus $*login* ir $*password* inicijuoja klasės metodą *reloadUser()*.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.26 lentelė Klasės CoreUser metodas __destruct()

public __destruct() – klasės destruktorius
Santrauka
void __destruct ()
Aprašymas
Klasės destruktorius. Inicijuoja tėvinės klasės *__destruct()* metodą.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.27 lentelė. Klasės CoreUser metodas singleton()

public singleton() – statinis metodas, kuris sukuria CoreUser klasės objektą
Santrauka
object _singleton ()
Aprašymas
Statiniam atributui *$_userInstance* priskiria naują *CoreUser* klasės objektą, jei jis dar nėra *CoreUser* klasės objektas.
Parametrai
Nėra
Grąžinama reikšmė
CoreUser klasės objektas

3.28 lentelė. Klasės CoreUser metodas __clone()

public __clone() – klasės kopijavimo metodas
Santrauka
void __clone ()
Aprašymas
Inicijavus šį metodą sukuriamas naujas *Exception CoreUserException* objektas.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.29 lentelė. Klasės CoreUser metodas __set()

public __set() – nustato nurodyto atributo iš $_data reikšmę
Santrauka
void __set (string $key, string $val)
Aprašymas
Jei nurodytas atributas *$key* egzistuoja masyve *$_data* priskiria jam *$val* reikšmę
Parametrai
string $key – masyvo *$_data* elemento vardas string $val – elemento *$key* iš masyvo *$_data* reikšmė
Grąžinama reikšmė
Nėra

3.30 lentelė. Klasės CoreUser metodas __get()

public __get() – gražina nurodyta masyvo *$_data* elementą
Santrauka
Midex __get (string $key)
Aprašymas
Grąžina masyvo *$_data $key* elemento reikšmę.
Parametrai
string $key – masyvo *$_data* elementas
Grąžinama reikšmė
Mixed

3.31 lentelė. Klasės CoreUser metodas reloadUser()

public realodUser() – perkrauna vartotojo duomenis
Santrauka
void reloadUser ()
Aprašymas
Jei *$_data* nėra masyvas arba tuščias masyvas inicijuoja klasės metodą *loadUser()*
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.32 lentelė. Klasės CoreUser metodas loadUser()

public loadUser() – prijungia vartotoją prie sistemos
Santrauka
void loadUser ()
Aprašymas
Jei objektas *$session* turi atributą $user_id arba abu atributus $*login* ir $*password* nuskaito vartotojo duomenis pagal nurodytus atributus iš duomenų bazės. Masyve *$_data* sukuria *user_id*, *user_login*, *user_email*, *user_name*, *user_registered*, *user_enabled*, *user_is_admin* elementus ir priskiria jiems pradines reikšmes *null*. Jei pagal anksčiau nurodytus parametrus vartotojas buvo rastas duomenų bazėje masyvo *$_data* elementams priskiria atitinkamas vartotojo duomenų reikšmes. Jei vartotojas gali prisijungti prie sistemos nustato atitinkamas objekto *$session user_id* ir *user_is_admin* reikšmes.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.33 lentelė. Klasės CoreUser metodas unloadUser()

public unloadUser() – atjungia vartotoją nuo sistemos
Santrauka
void realodUser ()
Aprašymas
Nustato visas masyvo *$_data* elementų reikšmes į *null* ir sunaikina objekto *$session* vartotojo atributus.
Parametrai
string $key – masyvo *$_data* elementas
Grąžinama reikšmė
Nėra

3.34 lentelė. Klasės CoreUser metodas getData()

public getData() – gražina klasės atributo *$_data* reikšmę
Santrauka
array _getData ()
Aprašymas
Gražina klasės atributo *$_data* reikšmę.
Parametrai
Nėra
Grąžinama reikšmė
Klasės atributo *$_data* reikšmė

3.3.7. Klasė CoreSessionException

3.35 lentelė Klasės CoreSessionException aprašymas

Aprašymas
CoreSession klasės klaidų valdymo klasė
Ryšiai su kitomis klasėmis
Apibendrinimo ryšys su PHP klase *Exception*.
Atributai
Nėra

3.3.8. Klasė CoreSession

3.36 lentelė Klasės CoreSession aprašymas

Aprašymas
Vartotojo sesijos informacijos valdymo klasė
Ryšiai su kitomis klasėmis
Nėra
Atributai
private $_sessionInstance – statinis pačios klasės objektas

3.37 lentelė Klasės CoreSession metodas __construct()

public __construct() – klasės konstruktorius
Santrauka
void __construct ()
Aprašymas
Inicijuoja PHP funckiją *session_start()*.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.38 lentelė Klasės CoreSession metodas singleton()

public singleton() – statinis metodas, kuris sukuria *CoreSession* klasės objektą
Santrauka
void singleton ()
Aprašymas
Statiniam atributui *$_sessionInstance* priskiria naują *CoreSession* klasės objektą.
Parametrai
Nėra
Grąžinama reikšmė
CoreSession klasės objektas

3.39 lentelė Klasės CoreSession metodas __clone()

public __clone() – klasės kopijavimo metodas
Santrauka
void __clone ()
Aprašymas
Inicijavus šį metodą sukuriamas naujas *Exception CoreSessionException* objektas.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.40 lentelė Klasės CoreSession metodas destroy()

public destroy() – sunaikina sesiją
Santrauka
void destroy ()
Aprašymas
Inicijuoja *session_start()* metodą ir sunaikina visus sesijos elementus.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.41 lentelė Klasės CoreSession metodas __set()

public __set() – nustato nurodyto atributo iš masyvo $_SESSION reikšmę
Santrauka
void __set (string $key, string $val)
Aprašymas
$_SESSION $key elementui priskiria *$val* reikšmę
Parametrai
string $key – masyvo *$_SESSION* elemento vardas string $val – elemento *$key* iš masyvo *$_SESSION* reikšmė
Grąžinama reikšmė
Nėra

3.42 lentelė Klasės CoreSession metodas __get()

public __get() – gražina nurodyta masyvo $_SESSION elementą
Santrauka
mixed __get (string $key)
Aprašymas
Grąžina masyvo *$_SESSION $key* elemento reikšmę.
Parametrai
string $key – masyvo *$_SESSION* elementas
Grąžinama reikšmė
Mixed

3.43 lentelė Klasės CoreSession metodas unsetVar()

public unsetVar() – sunaikina nurodytus $_SESSION elementus
Santrauka
void unsetVar (mixed $var)
Aprašymas
Sunaikina nurodytus *$_SESSION* elementus
Parametrai
mixed $var – vienas elementas arba elementų masyvas, kuriuos reikia sunaikinti
Grąžinama reikšmė
Nėra

3.3.9. Klasė CorePluginException

3.44 lentelė Klasės CorePluginException aprašymas

Aprašymas
CorePlugin klasės klaidų valdymo klasė.
Ryšiai su kitomis klasėmis
Apibendrinimo ryšys su PHP klase Exception.
Atributai
Nėra

3.3.10. Klasė CorePlugin

3.45 lentelė Klasės CorePlugin aprašymas

Aprašymas
Sistemos komponentų apibendrinimo klasė
Ryšiai su kitomis klasėmis
Apibendrinimo ryšys su *CoreWeb* klase
Atributai
private $_modules – sistemos komponentų sąrašas private $_settings – sistemos komponentų nustatymų sąrašas private $_control – komponentų veiksmų nurodymų masyvas private $_currentModule – nurodo, kuris komponentas šiuo metu yra pagrindinis private $_pluginMessage – klasės pranešimas

3.46 lentelė Klasės CorePlugin metodas __construct()

public __construct() – klasės konstruktorius
Santrauka
void __construct ([array $control = array()])
Aprašymas
Inicijuoja tėvinės klasės *__construct()* metodą. Jei klasės atributas *$_moduleControl* nėra masyvas arba yra tuščias masyvas priskiria jam *$control* reikšmę. Jei klasės atributas *$_settings* nėra masyvas arba yra tuščias masyvas priskiria jam klasės metodo *_loadSettings()* gąžinamą reikšmę. Jei klasės atributas *$_modules* nėra masyvas arba yra tuščias masyvas priskiria jam klasės metodo *_loadModules()* grąžinamą reikšmę.
Parametrai
array $control – komponentų valdymo veiksmų masyvas
Grąžinama reikšmė
Nėra

3.47 lentelė Klasės CorePlugin metodas __destruct()

public __destruct() – klasės destruktorius
Santrauka
void __destruct ()
Aprašymas
Inicijuoja tėvinės klasės *__destruct()* metodą.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.48 lentelė Klasės CorePlugin metodas setCurrentModule()

public setCurrentModule () – nustato klasės atributo *$_currentModule* reikšmę
Santrauka
void setCurrentModule (string $module)
Aprašymas
Nustato klasės atributo *$_currentModule* reikšmę lygią parametrui *$module*.
Parametrai
string $module – nurodo, kuris komponentas šiuo metu bus pagrindinis
Grąžinama reikšmė
Nėra

3.49 lentelė Klasės CorePlugin metodas getCurrentModule()

public getCurrentModule() – grąžina klasės atributo *$_currentModule* reikšmę
Santrauka
string getCurrentModule ()
Aprašymas
Grąžina klasės atributo *$_currentModule* reikšmę.
Parametrai
Nėra
Grąžinama reikšmė
Klasės atributo *$_currentModule* reikšmė

3.50 lentelė Klasės CorePlugin metodas getMessage()

public getMessage() – grąžina klasės atributo *$_pluginMessage* reikšmę
Santrauka
string getMessage ()
Aprašymas
Grąžina klasės atributo *$_pluginMessage* reikšmę.
Parametrai
Nėra
Grąžinama reikšmė
Klasės atributo *$_pluginMessage* reikšmė

3.51 lentelė Klasės CorePlugin metodas getControl()

public getControl() – grąžina klasės atributo *$_control* reikšmę
Santrauka
array getControl ()
Aprašymas
Grąžina klasės atributo *$_control* reikšmę.
Parametrai
Nėra
Grąžinama reikšmė
Klasės atributo *$_control* reikšmė

3.52 lentelė Klasės CorePlugin metodas getModules()

public getModules() – grąžina klasės atributo $_modules reikšmę
Santrauka
array getModules ()
Aprašymas
Grąžina klasės atributo *$_modules* reikšmę.
Parametrai
Nėra
Grąžinama reikšmė
Klasės atributo *$_modules* reikšmė

3.53 lentelė Klasės CorePlugin metodas _loadModules()

private _loadModules() – grąžina vartotojui prieinamų komponrntų sąrašą
Santrauka
array _loadModules ()
Aprašymas
Pagal vartotojo lygį gražina jam prieinamų sistemos komponentų sąrašą
Parametrai
Nėra.
Grąžinama reikšmė
Vartotojui prieinamų sistemos komponentų sąrašas

3.54 lentelė Klasės CorePlugin metodas getAllModules()

public getAllModules() – grąžina visų duomenų bazėje registruotų komponentų sąrašą
Santrauka
array getAllModules ()
Aprašymas
Grąžina visų duomenų bazėje registruotų komponentų sąrašą.
Parametrai
Nėra.
Grąžinama reikšmė
Visų sistemos modulių informacijos sąrašas

3.55 lentelė Klasės CorePlugin metodas getSettings()

public getSettings() – grąžina klasės atributo $_settings reikšmę
Santrauka
array getSettings ()
Aprašymas
Grąžina klasės atributo *$_settings* reikšmę.
Parametrai
Nėra
Grąžinama reikšmė
Klasės atributo *$_settings* reikšmė

3.56 lentelė Klasės CorePlugin metodas _loadSettings()

protected _loadSettings() – grąžina sistemos komponentų nustatymų sąrašą
Santrauka
array _loadSettings ()
Aprašymas
Grąžina sistemos komponentų nustatymų sąrašą.
Parametrai
Nėra
Grąžinama reikšmė
Sistemos komponentų nustatymų sąrašas

3.57 lentelė Klasės CorePlugin metodas updateSettings()

public updateSettings() – atnaujina sistemos komponentų nustatymus
Santrauka
void updateSettings ([array $array = array()])
Aprašymas
Jei duotasis parametras *$array* yra masyvas suformuoja atitinkamą sistemos komponentų nustatymų atnaujinimo užklausą ir įvykdo ją duomenu bazėje. Sėkmingai įvykdžius atnaujinimą klasės atributui *$_settings* priskiria klasės metodo *_loadSettings()* grąžinamą reikšmę.
Parametrai
array $array – sistemos komponentųnustatymai
Grąžinama reikšmė
Nėra

3.58 lentelė Klasės CorePlugin metodas getAvailableModules()

public getAvailableModules() – grąžina nurodytos direktoijos *$dir* direktorijas
Santrauka
array getAvailableModules ([string $dir = null])
Aprašymas
Jei nenurodyta *$dir* reikšmė priskiria jei sistemos konstantos *MODULES_DIR* reikšmę. Skaito visus iš eilės, išskyrus *manager*, katalogus iš katalogo *$dir* ir deda juos į masyvą.
Parametrai
string $dir – katalogas, kurio katalogus reikia grąžinti.
Grąžinama reikšmė
Katalogo *$dir* katalogų (išskyrus *manager* katalogą) sąrašas.

3.59 lentelė Klasės CorePlugin metodas _pluginExists()

private _pluginExists() – patikrina ar nurodytas komponentas egzistuoja duomenų bazėje
Santrauka
bool _pluginExists (string $plugin)
Aprašymas
Patikrina ar nurodytas komponentas *$plugin* egzistuoja duomenų bazėje
Parametrai
string $plugin – komponento pavadinimas
Grąžinama reikšmė
True – jei toks komponentas jau egzistuoja False – kitu atveju

3.60 lentelė Klasės CorePlugin metodas _readFile()

private _readFile() – nuskaito failą
Santrauka
mixed _readFile (string $file)
Aprašymas
Nuskaito failo *$file* informaciją į vieną eilutę ir pakeičia joje rastas simbolių eilutes #_*TABLE_PREFIX*_# į sistemos konstantos *DB_PREFIX* reikšmes. Jei nurodytas failas neegzistuoja sukuria naują *Exception CorePluginException* objektą.
Parametrai
string $file – duomenų failas
Grąžinama reikšmė
Iš *$file* failo duomenų suformuota eilutė. False – jei failas *$file* nerastas arba buvo tuščias

3.61 lentelė Klasės CorePlugin metodas installPlugin()

public installPlugin() – įdiegia naują komponentą į sistemą
Santrauka
bool installPlugin (string $plugin [, string $sqlFile = null])
Aprašymas
Jei nurodyras *$sqlFile* atributas inicijuoja klasės metodą *_readFile()* ir gautą rezultatą įvykdo duomenų bazėje. Į duomenų bazę įrašo nurodytą *$plugin* komponentą. Jei vartotojas neturi reikiamų teisių, komponentas *$plugin* jau egzistuoja sistemoje ar nenurodytas *$plugin* atributas, klasės atributui *$_pluginError* priskiriams atitinkamas klaidos pranešimas.
Parametrai
string $plugin – komponento pavadinimas string $sqlFile – SQL užklausų failas
Grąžinama reikšmė
True – jei toks $plugin komponentas sėkmingai įdiegtas False – jei klasės atributas $_pluginError turi kokią nors reikšmę

3.62 lentelė Klasės CorePlugin metodas uninstallPlugin()

protected uninstallPlugin () – pašalina nurodytą sistemos komponentą
Santrauka
bool uninstallPlugin (string $plugin [, string $sqlFile = null])
Aprašymas
Jei nurodyras *$sqlFile* atributas inicijuoja klasės metodą *_readFile()* ir gautą rezultatą įvykdo duomenų bazėje. Iš duomenų bazės ištrina nurodytą *$plugin* komponentą. Jei vartotojas neturi reikiamų teisių, komponentas *$plugin* sistemoje neegzistuoja ar nenurodytas *$plugin* atributas, klasės atributui *$_pluginError* priskiriams atitinkamas klaidos pranešimas.
Parametrai
string $plugin – komponento pavadinimas string $sqlFile – SQL užklausų failas
Grąžinama reikšmė
True – jei toks $plugin komponentas sėkmingai pašalintas False – jei klasės atributas $_pluginError turi kokią nors reikšmę

3.3.11. Klasė CorePresenterException

3.63 lentelė Klasės CorePresenterException aprašymas

Aprašymas
Klasės CorePresenter klaidų valdymo klasė
Ryšiai su kitomis klasėmis
Apibendrinimo ryšys su PHP klase *Exception*
Atributai
Nėra

3.3.12. Klasė CorePresenter

3.64 lentelė Klasės CorePresenter aprašymas

Aprašymas
MVC modelyje už atitinkamos *View* dalies sukūrima atsakinga klasė
Ryšiai su kitomis klasėmis
Nėra
Atributai
Nėra

3.65 lentelė Klasės CorePresenter metodas factory()

public factory() – sukuria nurodytą vartotojo sąsajos objektą
Santrauka
mixed factory ([string $type = ‘smarty’ [, string $style = ‚default‘]])
Aprašymas
Sukuria naują *$type* tipo sistemos informacijai atvaizduoti skirtą objektą nurodydamas jam, kad informaciją reikės atvaizduoti naudojant *$style* stiliu. Sukuria *Exception CorePresenterException* objektą jei nurodytas tipas *$type* nerastas arba negalėjo sukurti jo objekto.
Parametrai
string $type – sistemos informacijai atvaizduoti skirto objekto tipas string $style – nurodo kokiu stiliumi reikės atvaizduoti sistemos pateikiamą informaciją.
Grąžinama reikšmė
Naują *$type* tipo sistemos informacijai atvaizduoti skirtą objektą

3.3.13. Klasė PresenterBase

3.66 lentelė Klasės PresenterBase aprašymas

Aprašymas
Abstrakti, sistemoje naudojamų pateiktos informacijos atvaizdavimui skirtų klaisų bazinė klasė.
Ryšiai su kitomis klasėmis
Nėra
Atributai
protected $_data – duomenys, kuriuos reikės atvaizduoti protected $_content – perdirbtų duomenų eilutė.

3.67 lentelė Klasės PresenterBase metodas setData()

public setData() – atvaizdavimui skirtų duomenų priskyrimas
Santrauka
void setData ([array $data = array()])
Aprašymas
Klasės atributui *$_data* priskiria duomenis, kuriuos reikės atvaizduoti.
Parametrai
array $data – duomenys, kurius reikės atvaizduoti
Grąžinama reikšmė
Nėra

3.68 lentelė Klasės PresenterBase metodas getContent()

public setData() – gražina klasės atributo $_content reikšmę
Santrauka
void getContent ()
Aprašymas
Gražina klasės atributo *$_content* reikšmę.
Parametrai
array $data – duomenys, kurius reikės atvaizduoti
Grąžinama reikšmė
Klasės atributo *$_content* reikšmė

3.69 lentelė Klasės PresenterBase metodas display()

public display() – abstraktus klasės metodas
Santrauka
void display ()
Aprašymas
Abstraktus klasės metodas.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.70 lentelė Klasės PresenterBase metodas fetch()

public fetch() – abstraktus klasės metodas
Santrauka
void fetch ()
Aprašymas
Abstraktus klasės metodas.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.3.14. Klasė PresenterSmarty

3.71 lentelė Klasės PresenterSmarty aprašymas

Aprašymas
Sistemos pateikemai informacijai atvaizduoti skirta klasė.
Ryšiai su kitomis klasėmis
Agreguoja *Smarty* klasę (http://smarty.php.net)
Atributai
private $_smarty – Smarty objektas private $_path – šablonų saugojimo kelias

3.72 lentelė Klasės PresenterSmarty metodas __construct()

public __construct() – Klasės konstruktorius
Santrauka
void __construct (string $style)
Aprašymas
Klasės atributui *$_smarty* priskiria naują *Smarty* objektą ir nustato jo savybes.
Parametrai
string $style – nurodo stilų, kuriuo reikės atvaizduoti sistemos pateikiamą informaciją.
Grąžinama reikšmė
Nėra

3.73 lentelė Klasės PresenterSmarty metodas display()

public display() – duomenų atvaizdavimui skirtas metodas
Santrauka
void display ()
Aprašymas
Apdoroja tėvinės klasės atributo *$_data* saugomą informaciją ir gautą rezultatą atvaizduoja vartotojo naršyklės lange.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.74 lentelė Klasės PresenterSmarty metodas fetch()

public fetch() – duomenų apdorojimo metodas
Santrauka
void fetch ()
Aprašymas
Daro tą patį ka ir klasės metodas *display()*, tik gautus rezultatus ne grąžina vartotojo naršyklei, o priskiria juos tėvinės klasės atributui *$_content*.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.3.15. Klasė CorePage

3.75 lentelė Klasės CorePage aprašymas

Aprašymas
Klasė skirta sistemos pateikiamai informacijai saugoti. Saugo HTML puslapio Meta duomenis, šablonus, kuriuos reikės atvaizduoti ir duomenis, kurie bus priskirti šablonams.
Ryšiai su kitomis klasėmis
Nėra
Atributai
private $_error – klaidos atributas private $_created – ar puslapis jau sukurtas ar ne? private $_container – sistemos pateikiamų duomenų saugykla private $_style – informacijos atvaizdavimo stilius private $_meta – puslpaio meta duomenų saugykla private $_parts – puslpaio sudedamosios dalys private $_defaultMeta – pagal nutylėjima priskiriamų meta duomenų masyvas private $_templates – puslapio dalių šablonų masyvas

3.76 lentelė Klasės CorePage metodas __construct()

public __construct() – klasės konstruktorius
Santrauka
void __construct ([string $style = DEFAULT_STYLE])
Aprašymas
Klasės konstruktorius. Klasės atributui *$_style* priskiria *$style* reikšmę..
Parametrai
string $_style – stilius, kuriuo bus atvaizduojama sistemos pateikiama informacija.
Grąžinama reikšmė
Nėra

3.77 lentelė Klasės CorePage metodas addContent()

public addContent() – talpina informaciją į $_container atributą
Santrauka
void addContent (string $part, string $data)
Aprašymas
Jei klasės atributas *$_error* yra *false*, *$data* yra masyvas ir *$part* yra klasės atributo *$_parts* elementas prideda *$data* duomenis į klasės atributo *$_container[$part]* sekantį elementą.
Parametrai
string $part – nurodo į kurią *$_container* dalį pridėti *$data* duomenis array $data – duomenų masyvas
Grąžinama reikšmė
Nėra

3.78 lentelė Klasės CorePage metodas addMeta()

public addMeta() – meta duomenų pridėjimui skirtas metodas
Santrauka
void addMeta (array $data)
Aprašymas
Jei *$data* yra masyvas prijungia jį prie klasės atributo *$_meta*.
Parametrai
array $data – puslapio metaduomenų masyvas
Grąžinama reikšmė
Nėra

3.79 lentelė Klasės CorePage metodas _removeQuotes()

private _removeQuotes() – panaikina visas dvigubas kabutes
Santrauka
mixed _removeQuotes (mixed $data)
Aprašymas
Panaikina visas dvigubas kabutes
Parametrai
mixed $data – duomenys, kuriuose reikia panaikinti dvigubas kabutes
Grąžinama reikšmė
Eilutę su panaikintom dvigubom kabutėm – jei *$data* buvo eilutė Masyva su panaikintom dvigubom kabutėm iš jo elementų – jei *$data* buvo masyvas

3.80 lentelė Klasės CorePage metodas _createMeta()

private _createMeta() – sukuria puslapio meta duomenų masyvą
Santrauka
void _createMeta ()
Aprašymas
Sukuria būsimo puslapio metad duomenų masyvą *$_meta*. Jei klasės atributas *$_meta* nėra masyvas arba yra tuščias masyvas jam priskiria klasės atributo *$_defaultMeta* reikšmę.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.81 lentelė Klasės CorePage metodas _createPage()

public _createPage() – sukuria puslapio duomenų masyvą
Santrauka
void _createPage ()
Aprašymas
Papildo klasės atributą *$_container* reikiamais duomenimis, kad būtu galima atvaizduoti HTML puslapį. Nustato klasės atributo *$_created* reikšmę į t*rue*.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.82 lentelė Klasės CorePage metodas getContainer()

public getContainer() – grąžina klasės atributo *$_container* reikšmę
Santrauka
void getContainer ()
Aprašymas
Jei klasės atributo *$_created* reikšmė yra *false* inicijuoja klasės metodą *_createPage()*. Grąžina klasės atributo *$_container* reikšmę
Parametrai
Nėra
Grąžinama reikšmė
Klasės atributą *$_container*

3.3.16. Klasė CoreControl

3.83 lentelė Klasės CoreControl aprašymas

Aprašymas
MVC modelyje kontrolerio vaidmenį atliekanti klasė. Bazinė kontrolerio klasė, atliekanti Front kontrolerio vaidmenį.
Ryšiai su kitomis klasėmis
Agreguoja *CorePresenter*, *CorePage* ir *CorePlugin* klases
Atributai
private $_presenter – sistemos pateikiamos informacijos atvaizdavimui skirtas objektas private $_page – sistemos pateikiamos informacijos saugyklos objektas private $_plugin – sistemos modulių objektas private $_control – kontroliavimo komandų masyvas private $_style – stilius, kuriuo turės būti atvaizduota sistemos pateikiama informacija private $_presentationType – sistemos duomenims atvaizduoti skirto objekto tipas private $_module – šiuo metu vykdomo modulio pavadinimas

3.84 lentelė Klasės CoreControl metodas __construct()

public __construct() – klasės konstruktorius
Santrauka
void __construct ([array $request = array()])
Aprašymas
Klasės konstruktorius. Klasės atributui *$_control* priskiria parametro *$request* reikšmę. Nustato kitų klasės atributų reikšmes. Klasės atributui *$_presenter* priskiria sistemos duomenų atvaizdavimo objektą, atributui *$_page* – klasės *CorePage* objektą, atributui *$_plugin* – klasės *CorePlugin* objektą.
Parametrai
array $request – kontroliavimo komandų masyvas
Grąžinama reikšmė
Nėra

3.85 lentelė Klasės CoreControl metodas _addSlashes()

private _addSlashes() – truputi pakeista PHP *addslashes()* funkcija
Santrauka
mixed _addSlashes (mixed $data)
Aprašymas
Daro tą patį, ką ir PHP funkcija *addslashes()*, tik šią funckiją gali pritaikyti ir masyvui.
Parametrai
mixed $data – duomenų eilutė ar masyvas
Grąžinama reikšmė
Pakeista *$data* reikšmė

3.86 lentelė Klasės CoreControl metodas _stripSlashes()

private _stripSlashes() – truputi pakeista PHP *stripslashes()* funkcija
Santrauka
mixed _stripSlashes (mixed $data)
Aprašymas
Daro tą patį, ką ir PHP funkcija *stripslashes()*, tik šią funckiją gali pritaikyti ir masyvui.
Parametrai
mixed $data – duomenų eilutė ar masyvas
Grąžinama reikšmė
Pakeista *$data* reikšmė

3.87 lentelėKlasės CoreControl metodas _addContent ()

private _addContent () – perduoda duomenis klasės atributui *$_page*
Santrauka
void _addContent ()
Aprašymas
Paima duomenis iš visų sistemoje dalyvaujančių modulių ir perduoda juos klasės atributui *$_page*.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.88 lentelė Klasės CoreControl metodas process()

public process() – pagrindinis klasės metodas
Santrauka
void process ()
Aprašymas
Pagrindinis klasės metodas. Inicijuoja klasės metodą *_addContent()*, sistemos atvaizdavimo objektui *$_presenter* priskiria objekto *$_page* metodo *_getContainer()* rezultatą ir iškviečia objekto *$_presenter* metodą *_display()*.
Parametrai
Nėra
Grąžinama reikšmė
Nėra

3.4. Vidiniai komponentų standartai ir failų struktūra

Norint sukurti ir įdiegti naują komponentą pritaikytą karkasui, komponentas privalo atitikti šiuos reikalavimus:

1. Komponentą sudarantys failai turi būti patalpinti į vieną katalogą. Atskiri komponento falai gali būti grupuojami pakatalogiuose. Komponento katalogas turi būti pavadintas pagal komponento vardą (rekomenduojama vienaskaitos vardininkas, mažosiom raidėm) ir patalpintas sistemos *module* kataloge.

2. Kiekvienas komponentas pagrindiniame savo kataloge privalo turėti *module.control.php* failą, kuris atlieka komponento kontrolerio vaidmenį ir *block.control.php*, kuris atlieka komponento atvaizdavimo blokuose kontrolerio funkciją. Šiuose failuose turi būti klasės, atitinkamai pavadintos *Module{Komponento_vardas}Control* ir *Block{Komponento_vardas}Control*. Pavyzdžiui, *User* komponento kontrolerio klasės – *ModuleUserControl* ir *BlockUserControl*. Inicializuojant šias klases joms bus perduotas valdymo objektas, kurio pagalba bus galima prieiti prie duomenų bazės, vartotojo duomenų, vartotojo sesijos duomenų ir pan. Šios klasės privalo turėti visiems prieinamą metodą *getData()*. Metodas *getData()* privalo grąžinti duomenų masyvą tokiu formatu:

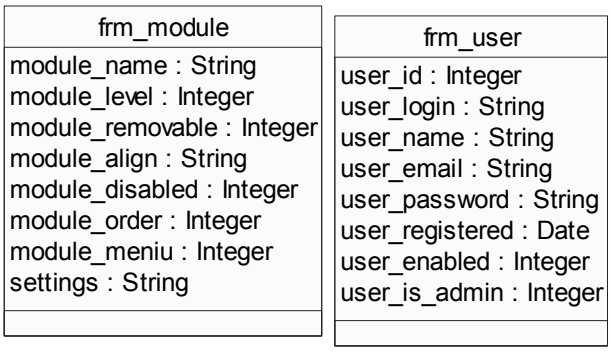
```
array (
„block“ => array (
                „dir“ => „kelias_iki_šablonų_katalogo“,
                „tpl“ => „šablono_failo_vardas“,
                „data“ => “duomenų, kurie bus priskirti šablonui, masyvas”
),
„module“ => array (
                „dir“ => „kelias_iki_šablonų_katalogo“,
                „tpl“ => „šablono_failo_vardas“,
                „data“ => “duomenų, kurie bus priskirti šablonui, masyvas”,
                „meta“ => array (
                        “title”=>“puslapio_pavadinimas”,
                        “js”=>”javascript failų masyvas”,
                        “name”=>array (
                                “keywords”=>“puslapio_raktažodžiai“,
                                “description”=>“puslapio_aprašymas“,
                                “robots”=>“robotų_komandos”,
                                ),
                        “http_equiv”=>array  (
                                “content-type”=>“text/html; charset=utf-8”,
                                ) ) )
```

Žemiau pateikiama failų struktūra, kad būtų galima įsivaizduoti karkaso ir komponentų išdėstymą.

```
|-index.php
|-install.sql
|-log.txt
|-settings.php
|-core\
|       |-control.class.php
|       |-db.class.php
|       |-object.class.php
|       |-page.class.php
|       |-plugin.class.php
|       |-presenter.class.php
|       |-reg.class.php
|       |-session.class.php
|       |-user.class.php
|       |-web.class.php
|       |-presentation\
|       |       |-base.class.php
|       |       |-smarty.php
|       |       |-Smarty\
|-module\
|       |-index.html
|       |-bottom\
|       |-default\
|       |       |-block.control.php
|       |       |-module.control.php
|       |       |-class\
|       |       |       |-default.class.php
|       |       |-manager\
|       |       |       |-manager.actions.php
|       |       |       |-manager.class.php
|       |       |-tpl\
|       |       |       |-default\
|       |       |       |       |-default.tpl
|       |       |       |       |-form.tpl
|       |-manager\
|       |       |-block.control.php
|       |       |-module.control.php
|       |       |-class\
|       |       |       |-managerblock.class.php
|       |       |       |-settings.class.php
|       |       |-tpl\
|       |       |       |-default\
|       |       |       |       |-modules_list.tpl
|       |       |       |       |-settings.tpl
|       |-menu\
|       |       |-block.control.php
|       |       |-class\
|       |       |       |-menublock.class.php
|       |       |-tpl\
|       |       |       |-default\
|       |       |       |       |-block.tpl
|       |-news\
|       |-top\
|       |-user\
|       |       |-block.control.php
|       |       |-module.control.php
|       |       |-class\
|       |       |       |-user.class.php
|       |       |       |-useradmin.class.php
|       |       |       |-userblock.class.php
|       |       |-tpl\
|       |       |       |-default\
|       |       |       |       |-ctrlpanel.tpl
|       |       |       |       |-list.tpl
|       |       |       |       |-login.tpl
|       |       |       |       |-message.tpl
|       |       |       |       |-password.tpl
|       |       |       |       |-registered.tpl
|       |       |       |       |-user_form.tpl
|-style\
|       |-default\
|       |       |-style.css
|-tpl\
|       |-default\
|       |       |-block_footer.tpl
|       |       |-block_header.tpl
|       |       |-page_404.tpl
|       |       |-page_bottom.tpl
|       |       |-page_center.tpl
|       |       |-page_end.tpl
|       |       |-page_error.tpl
|       |       |-page_footer.tpl
|       |       |-page_header.tpl
|       |       |-page_left.tpl
|       |       |-page_part.tpl
|       |       |-page_right.tpl
|       |       |-page_top.tpl
|       |       |-cache\
|       |       |-tpl_c\
```

3.5. Duomenų bazės schema

PHP komponentų karkasą sudaro tik 2 lentelės – frm_user ir frm_module. Visos kitos lentelės atsiranda instaliuojant papildomus komponentus. Juos pašalinus, lentelės taip pat pašalinamos iš duomenų bazės. PHP komponentą sudarančios lentelės su atributais ir atributų tipais pavaizduotos 3.11 paveiksle.

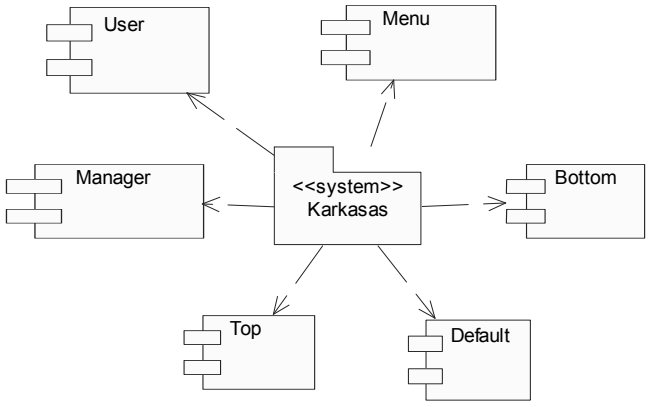


3.11 pav. Duomenų bazės schema

Atributai detaliai paaiškinti 2.2 skyriuje Karkaso duomenų struktūros.

3.6. Realizacijos modelis

3.6.1. Karkaso komponentinė architektūra

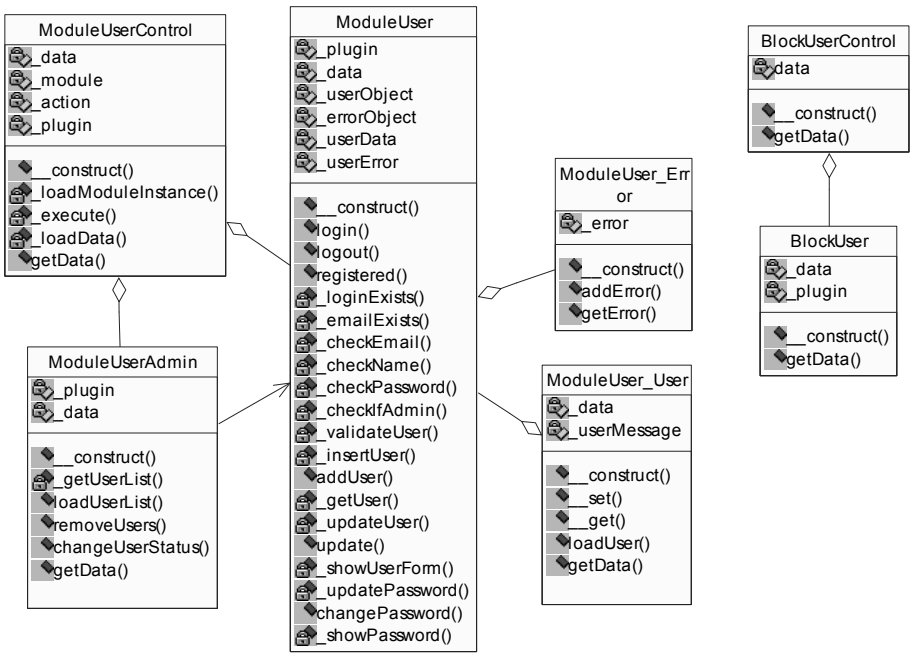


3.12 pav. Karkaso komponentinė architektūra

Į karkasą integruoti *User* (vartotojų valdymo), *Manager* (komponentų valdymo), *Top* (puslapio viršutinės dalies), *Bottom* (puslapio apatinės dalies, *Menu* (meniu) ir *Default* komponentai.

3.6.1.1. Komponentas *User*

Komponentas skirtas vartotojų informacijai valdyti. Glaudžiai siejasi su karkaso klase *CoreUser*. Šis komponentas valdo vartotojo registraciją bei prisijungimą prie sistemos. Komponento klasių diagrama pavaizduota 3.13 paveiksle.

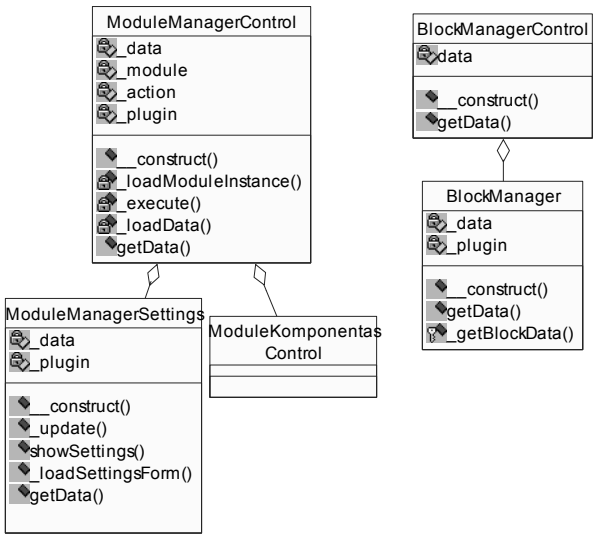


3.13 pav. Komponento *User* klasių diagrama

Klasė *ModuleUserControl* skirta valdyti šį komponentą. Priklausomai nuo pasirinkto metodo ji sukuria *ModuleUser* arba *ModuleUserAdmin* klasių objektus, kurie yra skirti vartotojo informacijai apdoroti. Klasė *ModuleUser* sukuria du papildomus objektus: klasės *ModuleUser_User* objektą, kuris yra skirtas duomenis apie vartotoją saugoti bei klasės *ModuleUser_Error* objektą, kuris yra skirtas vartotojo duomenimis manipuliavimo metu iškilusiems klaidų pranešimams saugoti. Klasė *BlockUserControl* skirta vartotojo komponento atvaizdavimo bloke valdymui. Ji sukuria klasės *BlockUser* objektą, kuris neprisijungusiam prie sistemos vartotojui grąžina prisijungimo, registracijos bei slaptažodžio priminimo formą, o prisijungusiam – savo duomenų keitimo bei atsijungimo formą.

3.6.1.2. Komponentas *Manager*

Komponentas skirtas sistemoje esančių komponentų valdymui. Glaudžiai susijęs su karkaso klase *CorePlugin*. Jo pagalba galima įdiegti naujus ar pašalinti iš sistemos jau esamus komponentus, taip pat inicijuoti komponentų informacijos atnaujinimo funkcijas. Komponento klasių diagrama pavaizduota 3.14 paveiksle.



3.14 pav. Komponento *Manager* klasių diagrama

Klasė *ModuleManagerControl* skirta valdyti šį komponentą. Jei vartotojas pasirinko sistemos komponentų tvarkymą, sukuriamas klasės *ModuleManagerSettings* objektas, kuris yra skirtas informacijai apie pačius komponentus atnaujinti, t.y. nustatyti ar komponentą rodyti sistemoje ar ne, ar įtraukti į meniu, nustatyti komponento bloko rodymo eilės tvarką bei puslapio pusę ir t.t. Jei vartotojas pasirenka kito komponento valdymą, sukuriamas to komponento kontrolerio klasės objektas, ir toliau visa veiksmų eiga priklauso nuo jo. Klasė *BlockManagerControl* skirta komponento atvaizdavimui bloke valdyti. Sukuria klasės *BlockManager* objektą kuris grąžina visų sistemoje įdiegtų ir neįdiegtų komponentų ir galimų su jais atlikti veiksmų sąrašą.

3.6.1.3. Kiti karkaso komponentai

Default - komponentas skirtas komponento pagal nutylėjimą parinkimui. Komponentas pagal nutylėjimą – tai toks komponentas, kuris bus rodomas, jei nebus parinktas joks kitas komponentas.

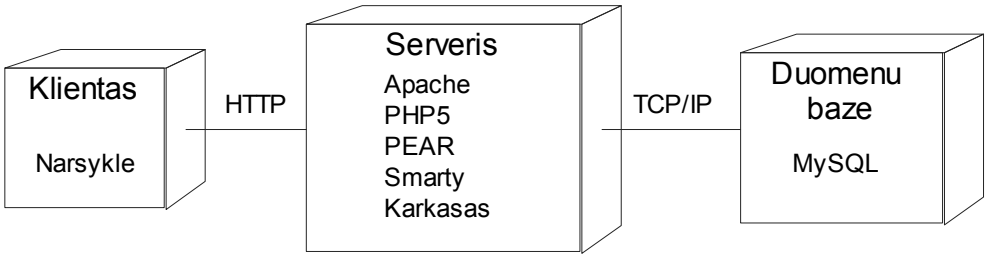
Menu – komponentas skirtas meniu, t.y. sistemoje esančių komponentų rodymui. Ar komponentą įtraukti į meniu, ar ne, galima nustatyti per *Manager* komponentą.

Top – puslapio viršutinės dalies kūrimo komponentas.

Bottom – puslapio apatinės dalies kūrimo komponentas.

3.6.2. Įdiegimo modelis

Įdiegimo modelį, pavaizduotą 3.15 paveiksle, sudaro klientas, serveris ir duomenų bazė. Klientas naudojasi interneto naršykle (Internet Explorer, Mozilla, Opera ir kt.) ir bendrauja su serveriu HTTP protokolo pagalba. Karkasas įdiegtas serveryje, kuriame turi būti Apache serverio programinė įranga, PHP5, PEAR, kuris įdiegiamas kartu su PHP5, ir šablonų vaizdavimo sistema Smarty, kuri yra įtraukta į karkasą. Serveris bendrauja su duomenų baze (MySQL) per TCP/IP protokolą.



3.15 pav. Įdiegimo modelis

3.7. Atskirų sistemos, sudarytos iš komponentų, dalių testavimo modelis

3.89 lentelė. „Naujo vartotojo įvedimas“ testavimo matrica

ID	Sąlyga	Prisijungimo vardas	Slaptažodis	Pakartotas slapta-žodis	El. paštas	Vartotojo vardas	Rezultatas (klaidos pranešimas)
TA1	Vartotojas neįvedė visai arba įvedė per trumpą prisijungimo vardą	Ne	Taip	Taip	Taip	Taip	Parodomas klaidos pranešimas
TA2	Vartotojas įvedė prisijungimo vardą turintį neleistinų simbolių	Ne	Taip	Taip	Taip	Taip	Parodomas klaidos pranešimas
TA3	Vartotojas įvedė sistemoje jau egzistuojantį prisijungimo vardą	Ne	Taip	Taip	Taip	Taip	Parodomas klaidos pranešimas
TA4	Vartotojas neįvedė visai arba įvedė per trumpą slaptažodį	Taip	Ne	Ne	Taip	Taip	Klaidos pranešimas apie būtinus laukus
TA5	Įvesti slaptažodžiai nesutampa	Taip	Taip	Ne	Taip	Taip	Klaidos pranešimas apie slaptažodžių nesutapimą
TA6	Vartotojas neįvedė el. pašto adreso	Taip	Taip	Taip	Ne	Taip	Klaidos pranešimas apie būtinus laukus
TA7	Vartotojas įvedė blogai suformatuota el. pašto adresą	Taip	Taip	Taip	Ne	Taip	Parodomas klaidos pranešimas
TA8	Vartotojas įvedė blogą elektroninio pašto adreso domeną	Taip	Taip	Taip	Ne	Taip	Parodomas klaidos pranešimas
TA9	Vartotojas įvedė sistemoje jau egzistuojantį el. pašto adresą	Taip	Taip	Taip	Ne	Taip	Parodomas klaidos pranešimas
TA 10	Vartotojas neįvedė visai arba įvedė per trumpą savo vardą	Taip	Taip	Taip	Taip	Ne	Klaidos pranešimas apie egzistuojantį vartotoją
TA 11	Vartotojas įvedė savo vardą turintį neleistinų simbolių	Taip	Taip	Taip	Taip	Ne	Parodomas klaidos pranešimas

4. EKSPERIMENTINIS KOMPONENTINIŲ INTERNETO SISTEMŲ KŪRIMO KARKASO DIEGIMAS

4.1. Eksperimentinio diegimo aprašymas

Komponentinių interneto sistemų kūrimo karkasas buvo įdiegtas WINDOWS ir LINUX aplinkoje.

1. Sistemos turinys buvo nukopijuotas į išoriniam vartotojui per naršyklę prieinamą vietą. Windows aplinkoje – Apache įdiegimo direktorijos *htdocs* subdirektoriją, o Linux aplinkoje – vartotojo public_html direktoriją.

2. Duomenų bazėje buvo įvykdytas failas *install.sql*. Įvykdžius failą duomenų bazėje buvo sukurtos dvi lentelės: *frm_module* ir *frm_user*. Lentelėje *frm_module* atsirado įrašai pavaizduoti 4.1 paveiksle. Lentelėje frm_user atsiro įrašas pavaizduotas 4.2 paveiksle.

module_name	module_level	module_removable	module_align	module_disabled	module_order	module_menu	module_noblock	module_settings
default	0	0	left	0	999	0	0	
manager	2	0	left	0	3	0	0	
menu	0	0	left	0	1	0	0	
user	0	0	left	0	2	0	0	

4.1 pav. Pradiniai lentelės *frm_module* įrašai duomenų bazėje

user_id	user_login	user_email	user_password	user_name	user_registered	user_enabled	user_is_admin
1	admin	admin@example.com	e00cf25ad42683b3df678c61f42c6bda	Admin	2006-01-07 15:45:17	1	1

4.2 pav. Pradinis lentelės *frm_user* įrašas duomenų bazėje

3. Buvo redaguotas *settings.php* failas:

a) *define('DIR_SEP', '\\')*; eilutę Linux aplinkoje buvo pakeista į *define('DIR_SEP', '/')*, Windows aplinkoje nieko nekeista.

b) *define('URL_PATH', 'http://localhost/')* eilutė nekeista, kadangi Apache serveris buvo įdiegtas kompiuteryje, kuris neturi savo antrojo lygio domeno vardo.

c) *define('DB_USER', 'dbuser');* eilutėje *dbuser* buvo pakeistas į duomenų bazės vartotojo vardą.

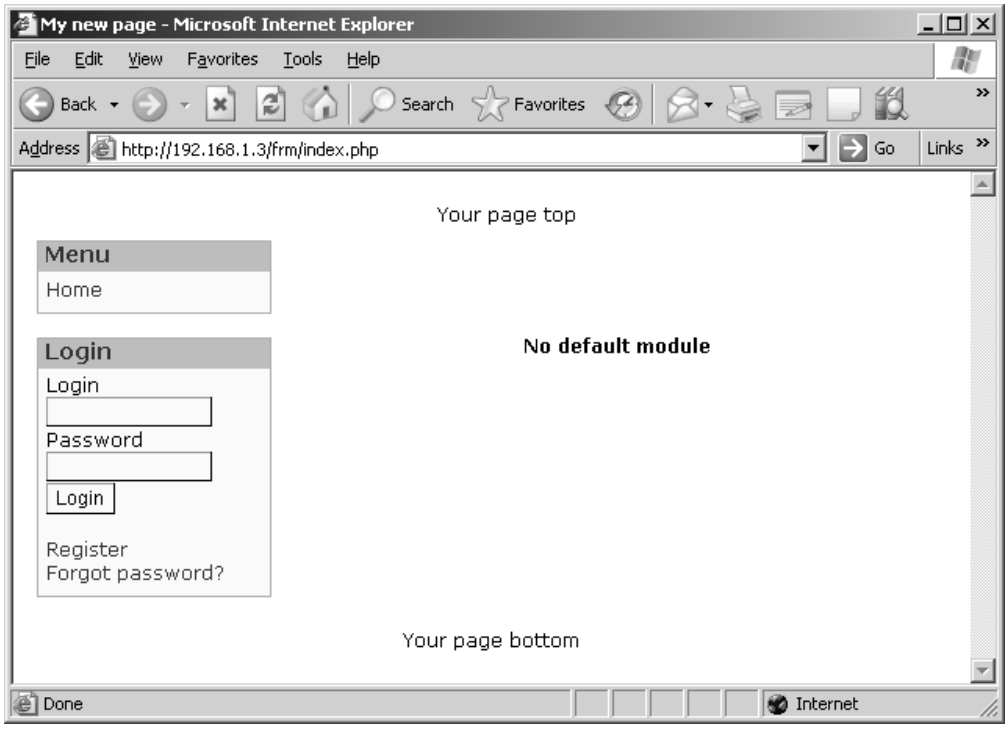
d) *define('DB_PASS', 'dbpassword')*; eilutėje *dbpassword* buvo pakeistas į duomenų bazės vartotojo slaptažodį.

e) *define('DB_HOST', 'localhost')*; eilutė nekeista, kadangi duomenų bazė veikė tame pačiame kompiuteryje kaip ir įdiegtas karkasas.

f) *define('DB_DB', 'database')*; eilutėje *database* buvo pakeistas į duomenų bazės vardą.

g) *define('DB_PREFIX', 'frm_')*; eilutė nekeista, nes priedėlis *frm_* yra tinkamas eksperimentiniam diegimui.

Eksperimento metu buvo naudojamos Internet Explorer ir Mozilla Firefox naršyklės. Abi naršyklės rodė vienodą vaizdą, parodytą 4.3 paveiksle.



4.3 pav. Pradinis eksperimentinio diegimo sistemos langas

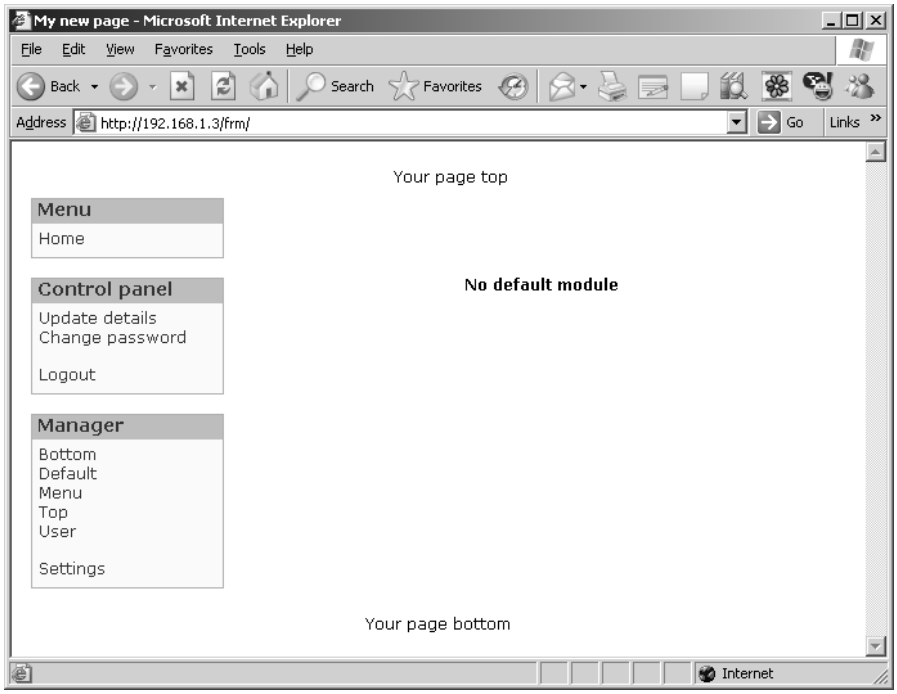
Eksperimentinio diegimo metu nebuvo gautas nė vienas klaidos pranešimas, todėl galima daryti išvadą, kad sistemos įdiegimas pavyko.

4.2. Komponentinių interneto sistemų kūrimo karkaso veikimo galimybės

Siekiant pademonstruoti komponentinių interneto sistemų kūrimo karkaso galimybes, aprašytas nedidelės interneto sistemos, kurioje pateikiamos naujienos, kūrimas.

4.2.1. Prisijungimas prie sistemos

Norint pradėti kurti naują interneto sistemą, pirmiausia reikia prie jos prisijungti, įvedant į atitinkamus prisijungimo bloko laukelius vartotojo vardą ir slaptažodį. Įdiegus sistemą, sukuriamas vartotojas su prisijungimo vardu *admin*, slaptažodžiu *admin1*ir turintis administratoriaus teises. Vartotojo duomenis, dėl saugumo, rekomenduojama pakeisti po pirmo prisijungimo. Prisijungus prie sistemos matome langą, pavaizduotą 4.4 paveiksle.

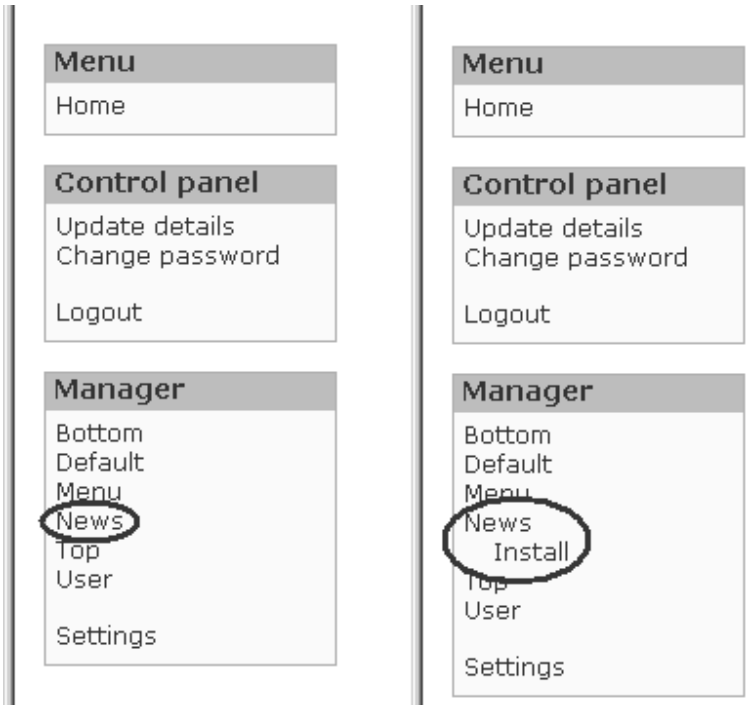


4.4 pav. Sistemos langas, administratoriui prisijungus prie sistemos

Karkase yra integruoti komponentai, kurie sudaro pradinį kuriamos sistemos pagrindą. Jie matomi *Manager* bloke.

4.2.2. Komponento *News* diegimas

Norint įdiegti naują komponentą, reikia jo turinį nukopijuoti į karkaso *module* direktoriją. Komponentas turi atitikti 3.4 skyriuje „Vidiniai komponentų standartai ir failų struktūra“ aprašytus standartus. Įkėlus iš anksto sukurtą komponentą ir atnaujinus naršyklės langą matosi, jog komponentas *News* atsirado *Manager* bloke (4.5 paveikslas, kairėje).



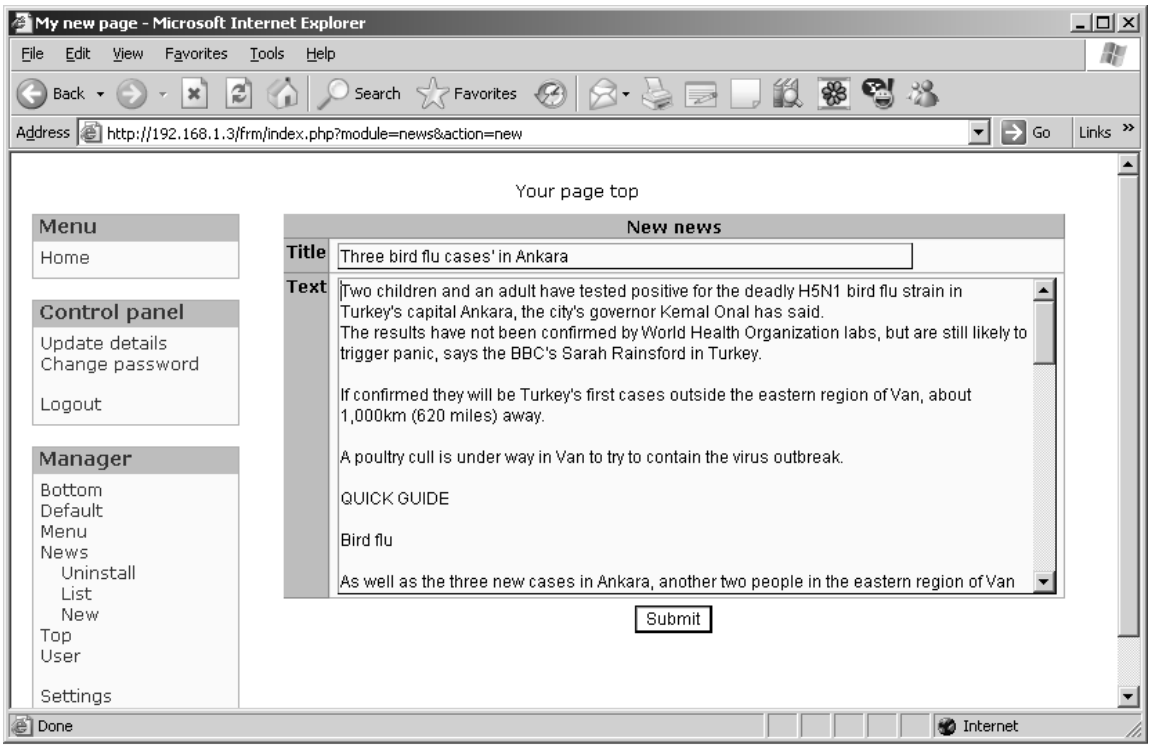
4.5 pav. Į karkasą įkeltas naujienų komponentas ir jo veiksmų sąrašas

Norint, kad komponentas atsirastų puslapyje, reikia jį instaliuoti. Paspaudus ant komponento *News*, parodomi galimi veiksmai ir pasirenkamas veiksmas *install* (4.5 paveikslas, dešinėje). Jei vidurinėje puslapio dalyje atsiranda užrašas „Module News was installed successfully“, komponentas News buvo įdiegtas sėkmingai. Atnaujinus sistemos langą, matosi, kad prie News komponento atsirado nauji galimi veiksmai ().

4.6 pav. Po komponento News įdiegimo atsiradę papildomi veiksmai

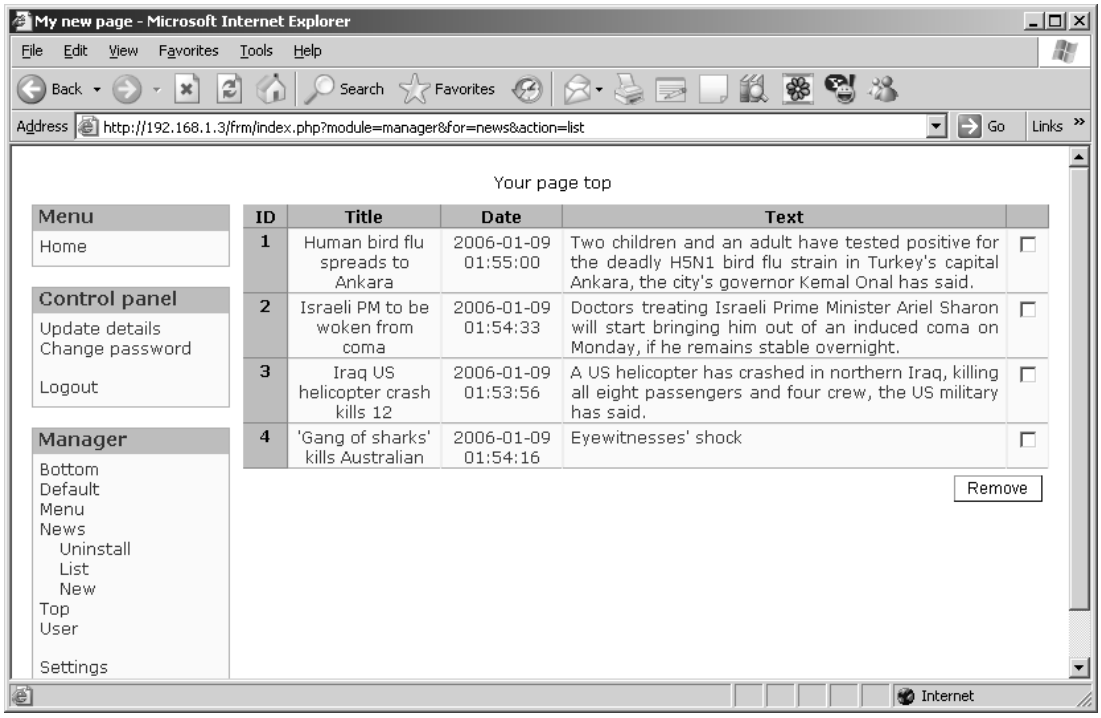
4.2.3. Komponento News informacijos tvarkymas

Prieš atvaizduojant naujienų komponentą sistemoje, reikia įvesti publikuojamas naujienas. Norint pridėti naujieną, reikia pasirinkti komponento *News* veiksmą *New*. Lango vidurinėje dalyje matome naujienų įvedimo formą, kurios atitinkamuose laukeliuose įvedame naujienos pavadinimą *title* ir naujienos tekstą (*text*), kaip pavaizduota 4.7 paveiksle.



4.7 pav. Naujienos priskyrimas komponentui News

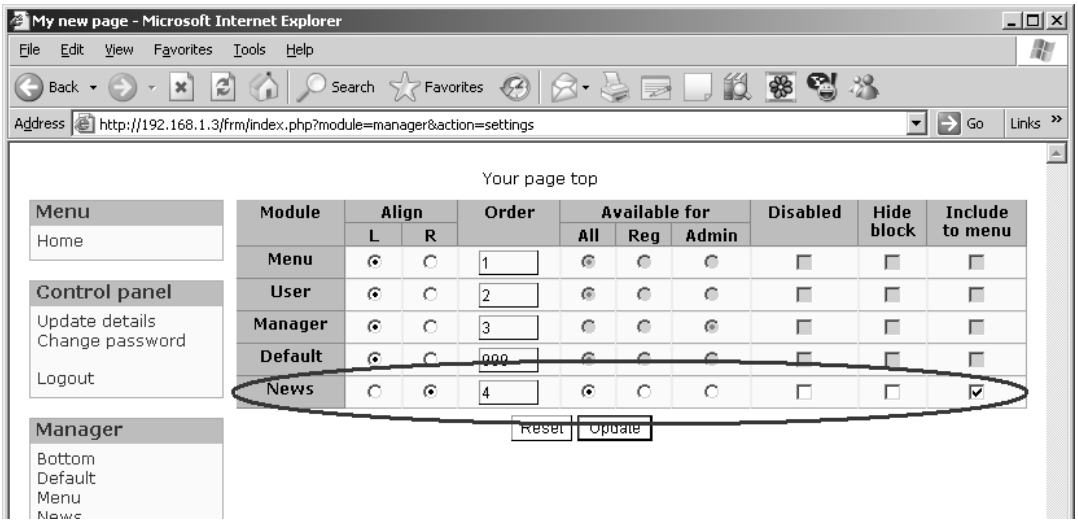
Analogiškai įdedame ir kitas naujienas. Pasirinkus veiksmą *List*, vidurinėje puslapio dalyje parodomas įdėtų naujienų sąrašas, pavaizduotas 4.8 paveiksle.



4.8 pav. Įdėtų naujienų sąrašas

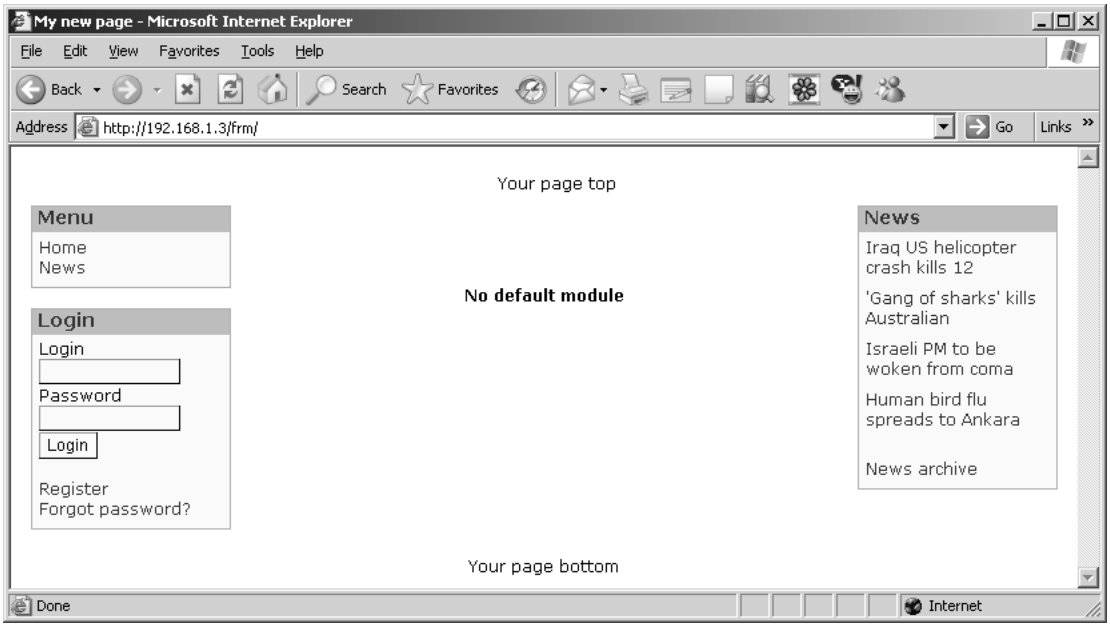
4.2.4. Komponento News atvaizdavimas puslapyje

Norint, kad komponentas *News* būtų atvaizduotas kuriamoje naujienų sistemoje, reikia atlikti tam tikrus nustatymus. Pasirinkus bloko *Manager* veiksmą *Settings*, vidurinėje puslapio dalyje matoma komponentų valdymo forma. Pažymime atitinkamus laukelius, kaip pavaizduota 4.9 paveiksle.



4.9 pav. Naujienų komponentų nustatymai

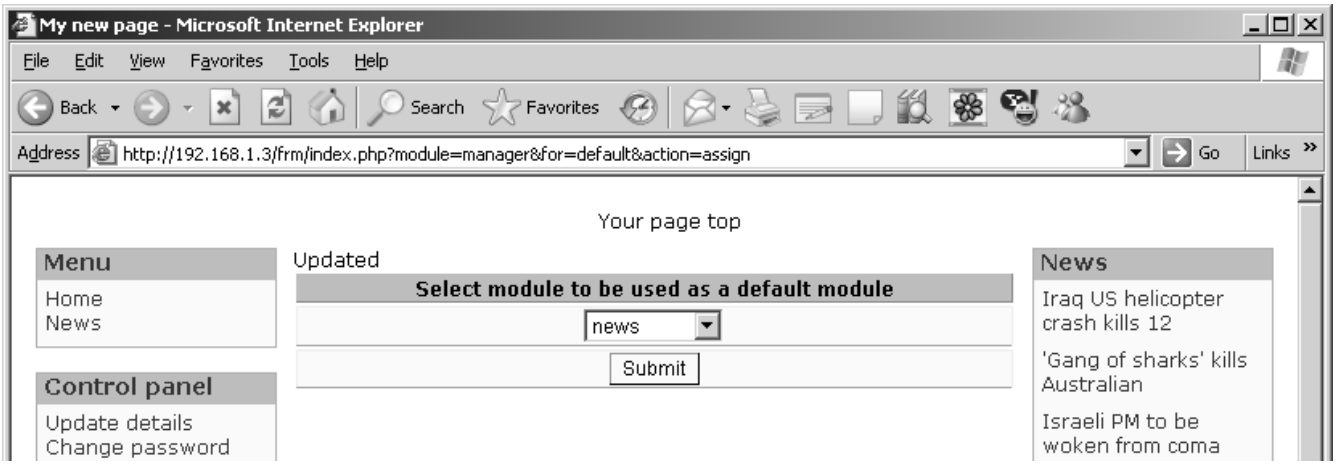
Šiuo atveju naujienų bloką News rodys dešinėje sistemos pusėje, jo eiliškumas 4, bus matomas visiems vartotojams ir įtrauktas kaip meniu punktas. Išsaugoję sistemos duomenis ir atsijungę iš administravimo panelės matome langą pavaizduotą 4.10 paveiksle.



4.10 pav. Naujienų komponento *News* atvaizdavimas sistemoje

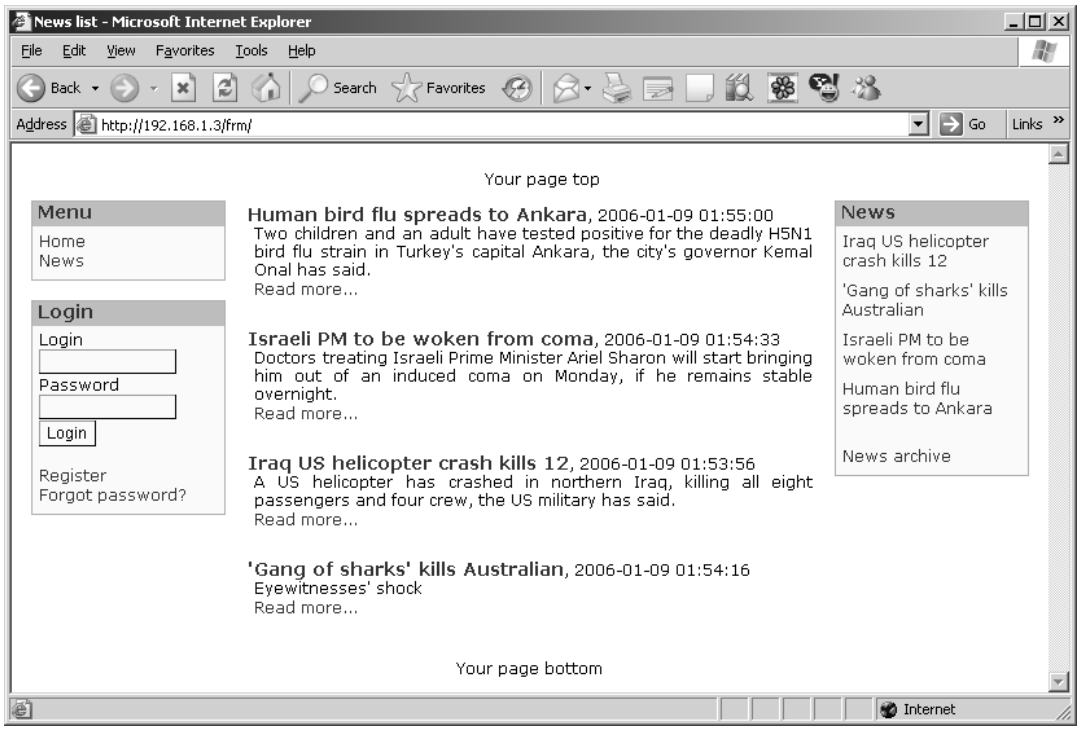
4.2.5. Komponentas Default

Norint užpildyti puslapio vidurinę dalį, komponentui *Default* reikia priskirti kito komponento informaciją. (Šiuo atveju naujienų komponentą *News*). Tuo tikslu vėl prisijungiama prie sistemos ir pasirenkamas bloko *Manager* komponento *Default* veiksmas *Assign*. Vidurinėje puslapio dalyje atsiradusioje formoje pasirenkamas naujienų komponentas *News* ir spaudžiama *Submit*, kaip pavaizduota 4.11 paveiksle.



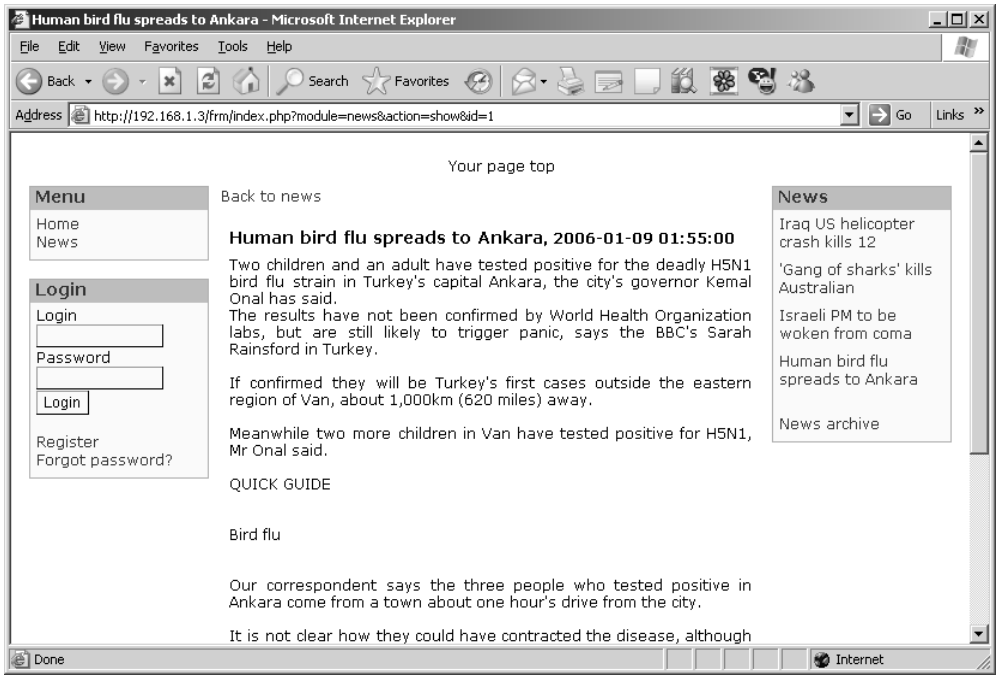
4.11 pav. Naujienų komponento priskyrimas vidurinei puslapio daliai

Atsijungę nuo sistemos matome, jog puslapio dalyje atsirado naujienų atvaizdavimas ().



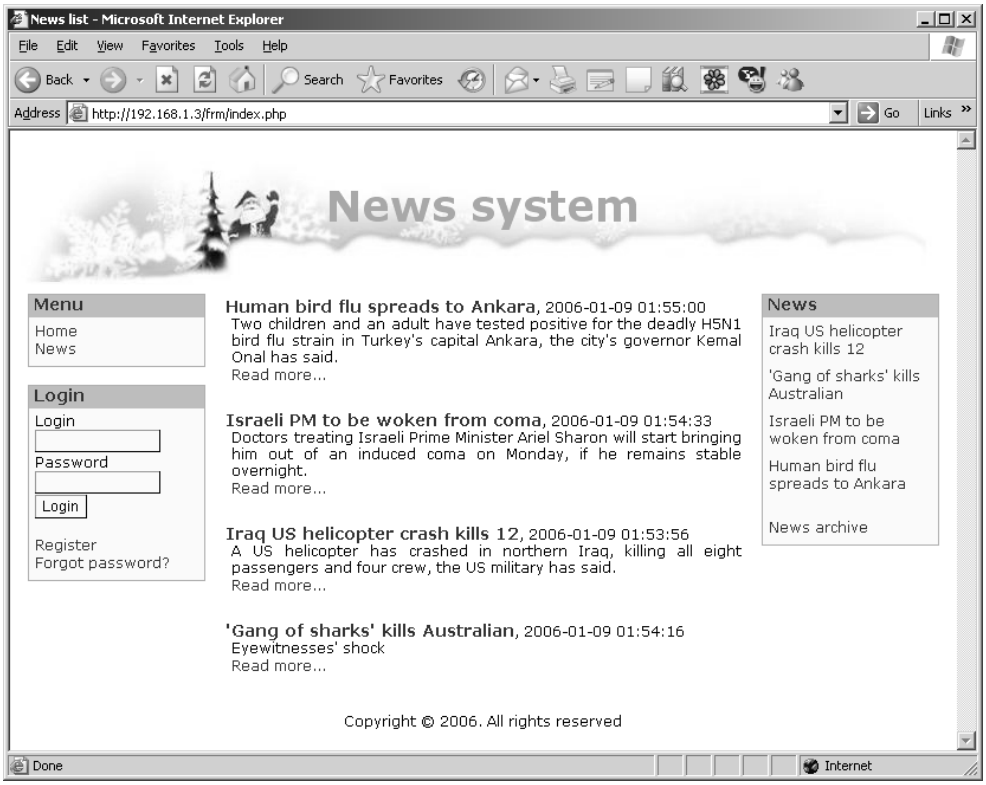
4.12 pav. Naujienų atvaizdavimas vidurinėje puslapio dalyje

Paspaudus ant atitinkamos naujienos pavadinimo arba ant nuorodos *Read more*, parodomas visas naujienos tekstas (4.13 paveikslas).



4.13 pav. Naujienos teksto atvaizdavimas

Analogiškai įdedami puslapio viršaus ir apačios komponentai *Top* ir *Bottom*, tačiau jų instaliuoti nereikia, nes jie jau integruoti į karkasą. *Top* komponente įdėjus fono paveiksliuką ir įrašius tekstą „News system“, o *Bottom* komponente įrašę tekstą „Copyright © 2006. All rights reserved” gauname nedidelę naujienų sistemą pavaizduotą 4.14 paveiksle.



4.14 pav. Sukurta naujienų sistema

Tokiai sistemai sukurti patyręs specialistas sugaištų 1-2 darbo dienas. Šio eksperimento metu sistema buvo sukurta per 15 minučių, nes komponentas *News* jau buvo sukurtas iš anksto. Tokio komponento kūrimas užimtų iki pusės darbo dienos. Galima daryti išvadą, kad kūrimo laikas naudojant karkasą smarkiai sumažėja.

4.3. Sistemos reikalavimų tenkinimo bei kokybės kriterijų įvertinimas ir savybių aprašymas

Norint įvertinti, ar realizuota sistema veikia tinkamai, atitinka kokybės kriterijus ir yra pranašesnė už sukurtas panašias sistemas, reikia įvertinti:

- Ar sistema atitinka išsikeltus reikalavimus?
- Ar sistema tenkina nurodytus kokybės kriterijus?
- Ar sistema turi nurodytas būtiniausias savybes?

4.3.1. Sistemos atitikimo reikalavimams įvertinimas

4.1 lentelėje pateikti pagrindiniai sistemos veikimui, duomenims bei funkcijoms išsikelti reikalavimai ir pagrįstas sistemos jiems atitikimas.

4.1 lentelė. Sistemos atitikimo reikalavimams įvertinimas

Reikalavimas	Atitikimo reikalavimui pagrindimas
Pateikiami duomenys turi būti griežtai standartizuoti tiek bendrai visai sistemai, tiek kiekvienam komponentui atskirai	Reikalavimas tenkinamas, nes sistema ir kiekvienas jos komponentas atitinka vidinius standartus aprašytus 3.4 skyriuje
Sistema turi dirbti stabiliai	Reikalavimas tenkinamas, nes komponentų keitimas, įdiegimas bei pašalinimas sistemos veikimui įtakos neturi.
Sistema turi būti patikima	Reikalavimas tenkinamas, nes sistema atlieka visus jai skirtus uždavinius.
Sistema turi būti lengvai plečiama	Reikalavimas tenkinamas, nes į sistemą galima lengvai įdiegti papildomus komponentus nesutrikdant bendro sistemos darbo ir tokiu būdu išplečiant jos funkcionalumą.
Naršyklei pateikiamas turinys turi atitikti XHTML standartus	Reikalavimas tenkinamas, nes remiantis W3C konsorciumo pateiktu validatoriumi [14], sistema atitinka XHTML standartus.
Vartotojų duomenų kiti matyti negali, nebent vartotojas tai leidžia	Reikalavimas tenkinamas, nes vartotojo duomenys yra apsaugoti slaptažodžiu, o pats slaptažodis sistemoje koduojamas.
Turi būti įgyvendintos visi analizės etape numatytos kompiuterizuojamos funkcijos	Sistemoje įgyvendintos analizės etape numatytos funkcijos: Peržiūrėti turinį, Užsiregistruoti, Prisijungti, Tvarkyti savo duomenis, Įdiegti/Ištrinti komponentą, Valdyti komponento informaciją, Valdyti komponentą, todėl reikalavimas tenkinamas.

Iš sistemos atitikimo reikalavimams įvertinimo matome, kad sistema atitinka visus jai išsikeltus reikalavimus.

4.3.2. Komponentinių interneto sistemų kūrimo karkaso savybių aprašymas lyginant su kitomis sistemomis

4.2 lentelėje pateikėme 1.6 skyriuje nagrinėtų sistemų ir suskurtos sistemos savybių palyginimą.

4.2 lentelė. Jau sukurtų sistemų palyginimas su mūsų sistema.

Sistemos savybės	PHP-Nuke	Xoops	Prado	Smart3	Mūsų sistema
Lengvai keičiamos sukurto puslapio vartojimo galimybės	+	+	?	?	+
Galimybė pačiam kurti ir integruoti naujus komponentus	+	+	+	+	+
Saugumas	-	+	?	?	+
Kuriant naudojamas objektinis programavimas	-	+	+	+	+
Kuriant naudojama PHP5	-	-	+	+	+
Pritaikytas MVC architektūrinis šablonas	-	-	-	-	+
Sistemos pateikiami duomenys atitinka XHTML standartą	-	+	-	?	+
Naudojami šablonai (templates)	-	+	-	+	+
Paprastas įdiegimas	+	+	+/-	+	+
Įdiegus sistemą, visos sistemos dalys funkcionuoja	-	-	+	+	+
Vartotojas neprivalo išmanyti PHP ar HTML norėdamas sukurti paprastą interneto sistemą	+	+	-	-	+
Nemokama	-	+	+	+	+

Pagal atliktą analizę matome, jog sukurta sistema turi visas teigiamas panašių sistemų savybes ir ištaiso jų trūkumus. Didžiausias komponentinių interneto sistemų kūrimo karkaso pranašumas lyginant su kitomis sistemomis yra tas, kad kuriant panaudotas MVC architektūrinis šablonas, kuris panaudotas kartu su CBSE sustiprina jo privalumus ir palengvina tolimesnį karkaso palaikymą.

4.3.3. Sistemos kokybės kriterijų įvertinimas

Įvertinant kokybės kriterijus buvo išrinkti 5 specialistai:

1. Giedrius Majauskas, projektų vadovas;

2. Sergejus Nosokas, projektų vadovas;

3. Aidas Berūgštis, programuotojas;

4. Simas Karbauskas, programuotojas;

5. Aurimas Grigas, vadybininkas.

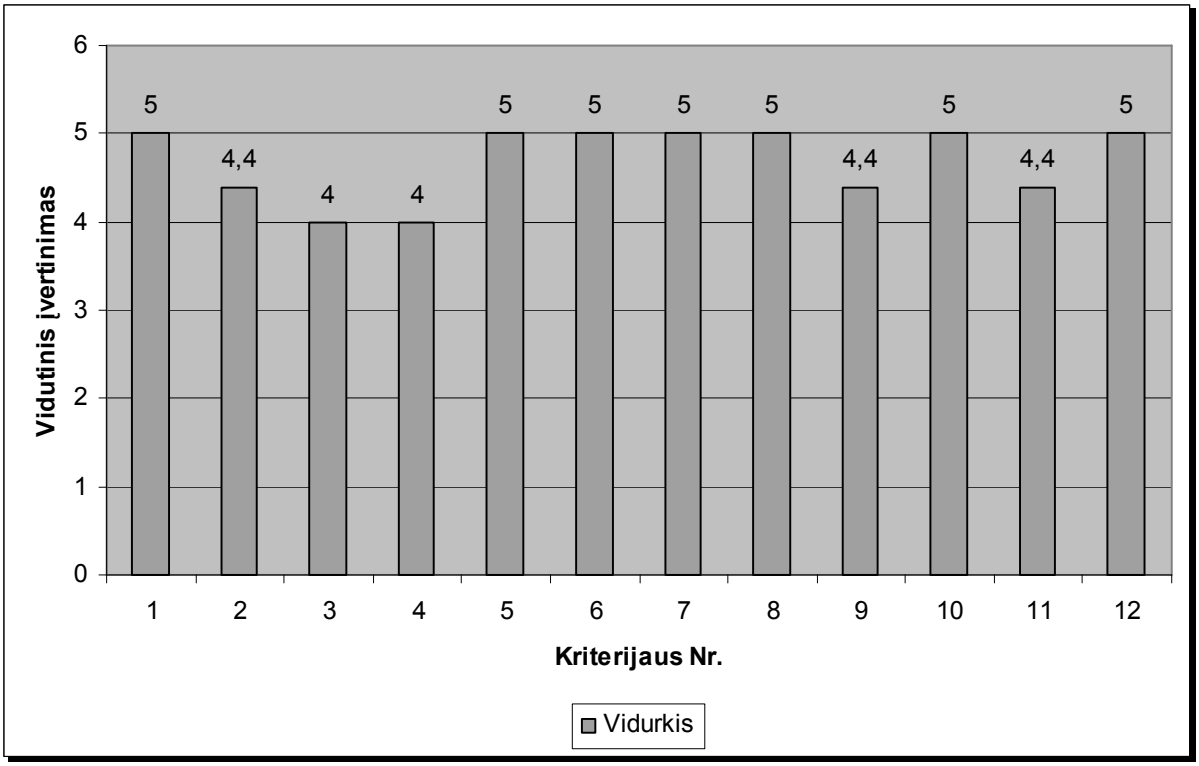
Buvo rašomas skaitinis įvertinimas - 1, 2, 3, 4 arba 5.

4.3 lentelė. Kokybės kriterijų įvertinimas

Eil. Nr.	Kokybės kriterijus	Vert. 1	Vert. 2	Vert. 3	Vert. 4	Vert. 5	Vidurkis
1	Ar sukurta sistema veikia tinkamai?	5	5	5	5	5	5
2	Ar išsprendžia išsikeltas problemas?	5	4	4	4	5	4,4
3	Ar sistema suprojektuota ir realizuota optimaliai?	4	3	4	4	5	4
4	Ar karkasą galima visiškai ar bent iš dalies pritaikyti įvairių sistemų kūrimui?	4	4	4	4	4	4
6	Ar sistema lengvai plečiama modifikuojant jau sukurtus komponentus arba pridedant naujus komponentus?	5	5	5	5	5	5
7	Ar panaikinti arba sumažinti rizikos faktoriai?	5	5	5	5	5	5
8	Ar sistema veikia be klaidų?	5	5	5	5	5	5
9	Ar bent minimaliai apsaugoti vartotojų asmeniniai duomenys?	4	5	4	4	5	4,4
10	Ar visi komponentai būtinai reikalingi?	5	5	5	5	5	5
11	Ar sistemą sudėtinga pritaikyti vartotojo reikmėms?	5	4	5	4	4	4,4
12	Ar sumažėjo sistemos kūrimo laikas naudojant komponentus?	5	5	5	5	5	5

Bendras sistemos įvertinimas - 4,6 iš 5.

Grafinis kokybės įvertinimas pavaizduotas 4.15 paveiksle.



4.15 pav. Kokybės kriterijų įvertinimas

Atlikus kokybės įvertinimą matome, kad sistema daugumą kriterijų tenkina. Silpniausiai įvertintas sistemos optimalumas ir pritaikymas įvairių sistemų kūrimui. Optimalumas įvertintas 4 balais todėl, kad ne visiškai aiški karkaso klasės *CoreObject* paskirtis, galima būtų pabandyti apsieiti ir be jos, tačiau bendram sistemos patikimumui tai įtakos neturi. Karkaso pritaikymas įvairių sistemų kūrimui įvertintas 4 balams todėl, kad informacijos atvaizdavimas puslapyje galėtų būti lankstesnis.

IŠVADOS

1. Sistemų sudėtingumo valdymui, laiko ir pinigų resursų mažinimui, tolimesniam sistemos palaikymui literatūroje buvo pateikti du metodai: komponentais pagrįstos programinės įrangos inžinerija (CBSE) ir šablonais paremta programinės įrangos architektūra (POSA). Išanalizavus šiuos metodus nustatyta, kad naudojant abu metodus kartu, jie duoda sinergetinį efektą.
2. Siekiant nustatyti reikalingas sistemos savybes buvo išnagrinėti ir palyginti pagal tam tikrus kriterijus panašūs karkasai (PHP-Nuke, Xoops, Smart3, Prado) ir išanalizuoti jų privalumai bei trūkumai. Didžiausias trūkumas tas, kad senai pradėtos kurti sistemos nesilaiko šiuolaikinių kodo rašymo standartų – tai apsunkina naujų komponentų kūrimą ir esamų integravimą į jau gyvuojančias sistemas. Taip pat nenaudojamas vienas iš literatūroje siūlomų metodų – POSA.
3. Remiantis panašių karkasų analize ir vartotojų problemomis, suprojektuotos pagrindinės kompiuterizuojamos sistemos funkcijos: peržiūrėti turinį, užsiregistruoti, prisijungti, tvarkyti savo duomenis, įdiegti/ištrinti komponentus, valdyti komponento informaciją, valdyti komponentus, valdyti sistemos vartotojus. Nubraižytos kiekvieno sistemos panaudojimo atvejo veiklos ir sekų diagramos, siekiant nustatyti karkaso funkcionalumą.
4. Remiantis išanalizuotais metodais, suprojektuota komponentinių interneto sistemų kūrimo karkaso loginė struktūra, atitinkanti MVC architektūrinį šabloną ir pakartotinio naudojimo komponentų principus.
5. Remiantis projektavimo etapo rezultatais, realizuotas komponentinių interneto sistemų kūrimo karkasas. Atliktas eksperimentinis sistemos diegimas ir sukurta bandomoji naujienų sistema. Laikas, skirtas sukurti sistemai naudojantis karkasu ženkliai sumažėjo.
6. Atlikus karkaso atitikimo reikalavimams įvertinimą nustatytą, kad karkasas atitinka visus iškeltus reikalavimus: pateikiami duomenys yra griežtai standartizuoti, sistema dirba stabiliai, yra patikima, lengvai plečiama, atitinka XHTML standartą, vartotojų duomenys bent minimaliai apsaugoti vartotojo vardu ir slaptažodžiu, įgyvendintos visos analizės etape nustatytos kompiuterizuojamos funkcijos.
7. Atlikus palyginimą su panašiais karkasais, pagal iškeltus svarbiausius tokios sistemos kriterijus galima daryti išvadą, kad realizuotas karkasas yra pranašesnis. Didžiausias pranašumas yra tai, kad suderinamos POSA ir CBSE metodų savybės, naudojamas PHP5 – objektinė programavimo kalba, vaizdavimas atskiriamas nuo programavimo naudojant šablonus.

8. Atlikus kokybės kriterijų įvertinimą pasiektas bendras sistemos naudingumo koeficientas – 4,6 iš 5 balų. Mažiausiai, po 4 balus, įvertinti sistemos optimalumas, nes ne visiškai aiški karkaso klasės CoreObject paskirtis ir karkaso pritaikymas įvairios paskirties sistemoms kurti, nes komponentų pateikiama informacija puslapyje atvaizduoja truputį nelanksčiai. Tačiau nustatyta, kad sistemos patikimumui tas įtakos neturi.

LITERATŪRA

1. ŠEINAUSKAS, Rimantas. *Pakartotinis panaudojimas projektavime* [interaktyvus]. Paskaitų medžiaga, Kauno Technologijos universitetas, 15 paskaita. [Žiūrėta 2004 m. spalio 10 d.]. Prieiga per internetą: <http://www.elen.ktu.lt/~rsei/SE4/index.htm>
2. CRNKOVIC, Ivica. Component–based Software Engeneering – new challanges in Sofware Development [interaktyvus]. [Žiūrėta 2005 m. spalio 12 d.]. Prieiga per internetą: <http://www.mrtc.mdh.se/publications/0328.pdf>
3. BARR, Paull C. *Component-Based Architecture Development* [interaktyvus],. [Žiūrėta 2004 m. gruodžio 27 d.]. Prieiga per internetą: <http://www.stc-online.org/cd-rom/1999/slides/SofComp8.pdf>
4. CHEESMAN, J. DANIELS J. *UML Components*, part I. [Adison Wesley], 2000,[Žiūrėta 2004 m. gruodžio 27 d.]
5. PHP-Nuke [interaktyvus], [Žiūrėta 2004 lapkričio 10 d.]. Prieiga per internetą: <http://www.phpnuke.org>
6. *XOOPS - eXtensible Object Oriented Portal System* [interaktyvus]. [Žiūrėta 2004 gruodžio 27 d.]. Prieiga per internetą: <http://www.xoops.org>
7. PRADO [interaktyvus], [Žiūrėta 2004-lapkričio 12 d.]. Prieiga per internetą:< http://www.xisc.com/>
8. *SMART3* [interaktyvus], [Žiūrėta 2001-lapkričio 12 d.]. Prieiga per internetą: <http://www.smart3.org/>
9. XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition) [interaktyvus] [žiūrėta 2004 gruodžio 28 d.]. Prieiga per internetą: < http://www.w3.org/TR/xhtml1/>
10. Cascading Style Sheets [interaktyvus] [žiūrėta 2004 gruodžio 28 d.]. Priega per internetą: <http://www.w3.org/Style/CSS/>
11. PUNSKYY Roman. Pattern Oriented Software Architecture [interaktyvus]. [Žiūrėta 2005 lapkričio 28 d.]. Prieiga per internetą:< http://www.cs.ucl.ac.uk/staff/W.Emmerich/lectures/3C05-02-03/aswe14-essay.pdf>
12. Model-View-Controller Architecture [interaktyvus]. [Žiūrėta 2005-12-01]. Prieiga per internetą: <http://rd13doc.cern.ch/Atlas/Notes/004/Note004-7.html>
13. PETROŠIŪTĖ Viktorija, SIMUTIS Algirdas. *Komponentinės architektūros pritaikymas PHP aplinkoje, internetines sistemas kuriančioje įmonėje*. INFORMACINĖS TECHNOLOGIJOS. 10-

toji tarpuniversitetinė magistrantų ir doktorantų konferencija. Konferencijos pranešimų medžiaga Technologija, Kaunas 2005, p. 94-97.

14. W3C Markup Validation Service [interaktyvus]. [Žiūrėta: 2006-01-07]. Prieiga per internetą: <http://validator.w3.org/>

TERMINŲ IR SANTRUMPŲ ŽODYNAS

1. CBSE (Component-based software engeneering) – komponentais grįstos programinės įrangos architektūra.

2. POSA (Pattern-Oriented software architecture) – Į šablonus orientuotos programinės įrangos architektūra.

3. MVC (Model-View-Controller) – POSA metodo architektūrinis šablonas.

4. Komponentas - tai programinės įrangos struktūrinis vienetas, kuris remiasi tam tikrais specifiniais principais.

5. Šablonas (Pattern) – surastas optimaliausias sprendimas nuolat pasikartojančioms problemoms spręsti programinės įrangos kūrime ir vystyme.

6. Šablonas (Template) – aibė iš anksto aprašytų formatų tekstui ir grafikai, naudojama interneto puslapiams kurti.

1 PRIEDAS. „Komponentinės architektūros pritaikymas PHP aplinkoje, internetines sistemas kuriančioje įmonėje“