KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Mindaugas Rimkus

Internetinių aplinkų kūrimo modelis panaudojant šablonus ir duomenų abstrakcijas

Magistro darbas

Darbo vadovas
dr. doc. Rimantas Butleris

KAUNAS, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMOS KATEDRA

Mindaugas Rimkus

Internetinių aplinkų kūrimo modelis panaudojant šablonus ir duomenų abstrakcijas

Magistro darbas

Kalbos konsultantė

Lietuvių kalbos katedros lektorė
dr. J. Mikelionienė

2006 05 29

Vadovas

dr. doc. Rimantas Butleris

2006 05 29

Recenzentas

dr. doc. E. Karčiauskas

2006 05 29

Atliko

IFM-0/2 gr. stud.
Mindaugas Rimkus

2006 05 29

KAUNAS, 2006

TURINYS

1 ĮŽANGA

Žiniatinklio (*web*) aplikacijų kūrimas praėjo ilgą raidos kelią, pradedant nuo HTML atsiradimo ir puslapių kodavimo be papildomų įrankių. Pirmosios žiniatinklio technologijos buvo nepajėgios generuoti puslapių pagal vartotojo pasirenkamas nuostatas. Duomenų bazėmis paremti tinklapiai kuriami nuskaitant duomenis ir sukuriami statiniai puslapiai, kurie sujungiami į navigacinę sistemą [1]. Toks sprendimas yra griozdiškas, jei informacija duomenų bazėje pasikeisdavo, tuomet reikėdavo visą žiniatinklio aplikaciją sugeneruoti iš naujo.

Plėtojantis serverio tipo programinei įrangai, atsirado technologijos CGI, ASP, PHP ir daug kitų. Jų dėka išankstinis žiniatinklio puslapių generavimas tapo technologine relikvija. Galimybė puslapius generuoti pagal konkrečią užklausą, leido sukurti visavertę žiniatinklio aplikaciją, kuri sąveikauja su vartotoju. Iš duomenų bazės įmanoma išgauti duomenis pagal tam tikrą kontekstą ir sukurti HTML atsakymą. Puslapių kūrimas pasidarė nepriklausomas nuo duomenų. Jų kaita nedaro įtakos žiniatinklio aplikacijos rekonstravimo problemoms.

Žiniatinklio aplikacijų generavimas pagal tam tikrą kontekstą (dinaminis puslapių kūrimas) panaudojant naujas žiniatinklio technologijas iškėlė kitą problemą. Puslapiai sukoduoti kalbomis C, CGI, Java Servlet talpina didžiulius HTML kodo įterpinius arba HTML dokumente PHP, ASP ir kitus programinio kodo segmentus. Šis programinio kodo ir HTML maišymas įtakoja vis didėjantį žiniatinklio puslapių projektavimo, kodavimo sudėtingumą [2]. Tokių puslapių rekonstravimas tampa sudėtinga rutina, kuri užima didelę laiko dalį.

Verslo logikos ir atvaizdavimo atskyrimo problema šiandien išsprendžiama šablonų pagalba. Naudojant šablonus, puslapio dizaineris gali kurti puslapio grafinį vaizdą įterpdamas kintamojo žymę, kurią programuotojas panaudoja aktyvuodamas puslapį. Pastaruoju metu šablonų sistemos yra labai populiarios ir naudojamos beveik visuose sudėtingesniuose projektuose.

Šiame darbe bus pateikta atviro kodo PHP programavimo kalbai skirtų šablonų sistemų taksonomija, analizė ir pasiūlytas šablonų sistemos modelis, kuris ypatingas tuo, kad įgalina atskirti ne tik logiką nuo atvaizdavimo, bet ir duomenis. Pasinaudojus pasiūlytu šablonų sistemos modeliu, atliksime projektą „STUDENTŲ PRAKTIKOS DARBŲ PORTALAS“. Pagrindinis projekto tikslas – studentų praktikos darbų pasirinkimo, pasiūlos, įdarbinimo, kontrolės palengvinimas. Kaip parodė dėstytojų apklausa šiuo metu praktikos darbų pasirinkimas ir kontrolė nėra trivialus dalykas. Dėstytojai ir studentai susiduria su problemomis: nedidelis firmų aktyvumas įdarbinimo procese, maža norimo darbo pasirinkimo

galimybė, maža studentų kontrolė. Pagrindinės funkcijos, kurias turi užtikrinti sukurtas portalas:

- Palengvinti ir automatizuoti praktikos darbų pasirinkimą. Būsimieji vartotojai – studentai;
- Palengvinti ir automatizuoti praktikos darbų valdymą. Būsimieji vartotojai – dėstytojai;
- Suteikti galimybes įmonės atstovams pasiūlyti praktikos darbus. Padėti jiems susirasti kvalifikuotą darbo jėgą;
- Sistema turi suteikti galimybes sistemos vartotojams bendrauti tarpusavyje;
- Sistema turi užtikrinti įvairių lygių priėjimo prie duomenų teises.

Šio portalo eksperimentiniame įvertinime pateiksime jo branduolio – šablonų sistemos Smarty [3] ir duomenų abstrakcijos klasės PEAR::DB [4] našumo charakteristikas. Atlikus eksperimentus, gautuose rezultatuose matyti, kad našiausia iš eksperimente naudotų šablonų sistemų yra Smarty. Šie rezultatai paaiškinami tuo, kad Smarty šablonų sistemoje galima panaudoti papildomas našumą spartinančias priemones. Naudojant duomenų abstrakcijos klasę PEAR::DB, eksperimentuose pastebėta našumo praradimas, lyginant ją su gimtosiomis PHP kalbos prieigos prie duomenų bazių funkcijomis, tačiau portatyvumo savybės ir objektiškai orientuota sąsaja, šios nežymius našumo praradimus kompensuoja su kaupu.

2 ŽINIATINKLIO (*WEB*) ŠABLONŲ TAKSONOMIJA IR ANALIZĖ

Kadangi egzistuoja daug įvairių žiniatinklio aplikacijų kūrimui skirtų šablonų sistemų, visas išanalizuoti būtų labai sunku, todėl egzistuojančias šablonų sistemas suskirstysime į tam tikras grupes. Pirmąją arba paprasčiausią klasifikacinę grupę sudarysime iš paprasčiausių šablonų sistemų, tiksliau programavimo stiliaus. Antrąją grupę - iš trečių šalių sukurtų šablonų sistemų, kuriose realizuotas tik paprasčiausias šablonų kintamųjų pakeitimas tam tikra informacija, kurią nustato programuotojas. Dizaineris yra atsakingas už tekstinio šablono sukūrimą ir kintamųjų pažymėjimą. Trečios grupės šablonų sistemos turi būti sudarytos iš visų savybių, kurias turi antros grupės šablonų sistemos, pridedant reikalavimus šablonų kompiliavimui bei papildomų modulių palaikymui. Ketvirta grupė bus sudaroma iš XML transformacijų pagrindu veikiančių šablonų sistemų, kurios įgyvendinamos XSLT pagalba. Paskutinę grupę sudarysime iš struktūrinių šablonų sistemų, kurios susideda iš trijų pagrindinių komponentų modelio, vaizdo ir valdiklio. Modelis atsakingas už priėjimą prie duomenų. Vaizdas – žiniatinklio puslapio grafinė pusė, dizainas. Valdiklis yra atsakingas už užklausų valdymą, susieja modelį ir vaizdą, suformuluoja atsakymą vartotojui.

Sudarome tokias žiniatinklio šablonų sistemų grupes:

1. Standartinėmis priemonėmis paremtas šablonų įgyvendinimas;
2. Paprastos šablonų sistemos;
3. Šablonų sistemos, palaikančios kompiliavimą ir papildomus modulius;
4. XSLT transformacijomis paremta šablonų sistema;
5. Modelis – Vaizdas – Valdiklis struktūrinės šablonų sistemos.

2.1 Standartinėmis priemonėmis paremtas šablonų įgyvendinimas

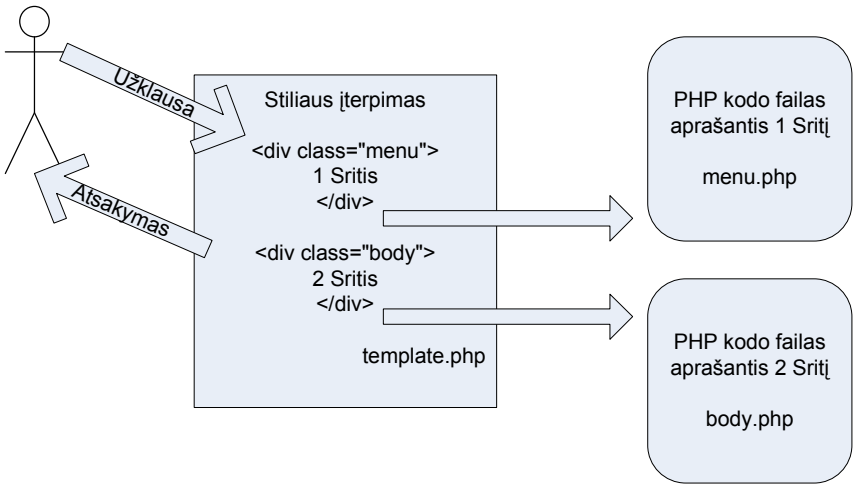
Standartinėmis priemonėmis paremtos šablonų sistemos yra nefunkcionalios ir netenkina logikos ir dizaino atskyrimo reikalavimų [5]. Šioje dalyje aptarsime siūlomus metodus kaip standartinėmis priemonėmis atskirti puslapio stilių, turinį ir struktūrą PHP programavimo kalboje.

2.1.1 Stiliaus, turinio ir struktūros atskyrimas

Stiliaus nuo turinio atskyrimui tradiciškai naudojamos CSS kaskadinių stilių lentelės ir XHTML praplėsta hiperteksto žymėjimo kalba, kuri suderima su XML. CSS dėka galima lengvai atnaujinti puslapio stilių ar susikurti keletą stilių. CSS formavimui panaudojus PHP galima žengti pirmą žingsnį link modulinės sistemos, kurią realizavus bus lengviau

atnaujinamas ne tik stilius, bet ir struktūra [6]. Iš esmės galima sukurti savo, labai riboto pobūdžio turinio valdymo sistemą.

Pagrindinis šablonas bus sudarytas iš XHTML, kuris skirtas struktūriniam žymėjimui, CSS atsakingas už turinio stilių ir PHP atliks šio proceso valdymą (1 pav.).

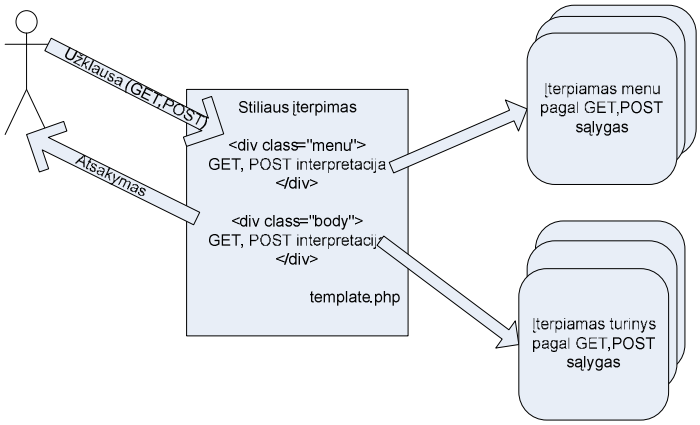


1 pav. Statinė standartinėmis priemonėmis paremtos šablonų sistemos realizacija

Pasinaudojant GET, POST metodais ir PHP, galima panaudoti kelias stilių lenteles bei importuoti skirtingus meniu ar turinio dokumentus tame pačiame šablone.

2.1.2 Turinio pasirinkimas naudojant PHP kintamuosius

Lig šiol aptartas šablono struktūros atskyrimas nuo turinio bei turinio įterpimas į šablono struktūrą aprašė tik statišką puslapį. Naudojant PHP kintamuosius iš vieno struktūros šablono failo įmanoma pasirinkti skirtingus turinio failus (2 pav.).



2 pav. Dinaminė standartinėmis priemonėmis paremtos šablonų sistemos realizacija

Pasinaudojant šia rekomendacija, puslapio struktūrai keisti užtenka tik vieną struktūrą, aprašantį dokumentą ir struktūros pakeitimai bus matomi visame tinklapyje.

Dažnai šiuos metodologijos maišomos tarpusavyje. Pavyzdžiui, puslapio meniu būna statinis, tuomet jis be jokių papildomų sąlygų įtraukiamas į pagrindinį struktūros šabloną ir pagal meniu sąlygą (GET metodą) parenkamas turinio failas.

Statinis puslapio įterpimas PHP kalboje pateiktas (3 pav.).

```
<div class="body">
<?php @ require_once ("page.html"); ?>
</div>
```

3 pav. Statinis puslapio įterpimas į struktūros šabloną

Dinaminis puslapio įterpimas PHP kalboje pateiktas (4 pav.).

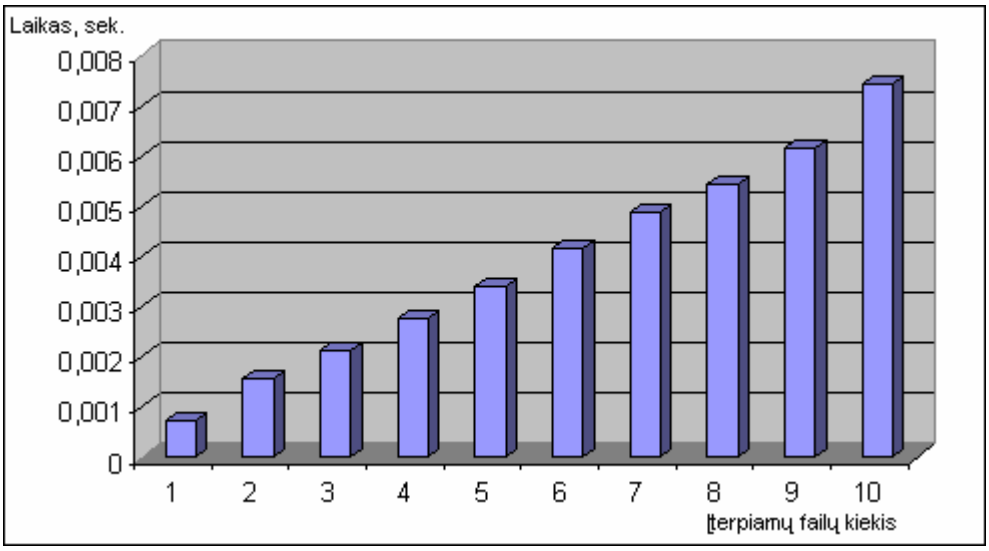
```
<div class="body">
<?php @ require_once ("$page.html"); ?>
</div>
```

4 pav. Dinaminis puslapio įterpimas į struktūros šabloną

PHP kalbos kintamieji yra identifikuojamai $ (dolerio) ženklu. Todėl pakeitus eilutę „page.html“ į „$page.html“ pasikeičia jos reikšmė. „$page.html“ yra pakeičiama $page kintamojo reikšme pridedant simbolių eilutę „.html“.

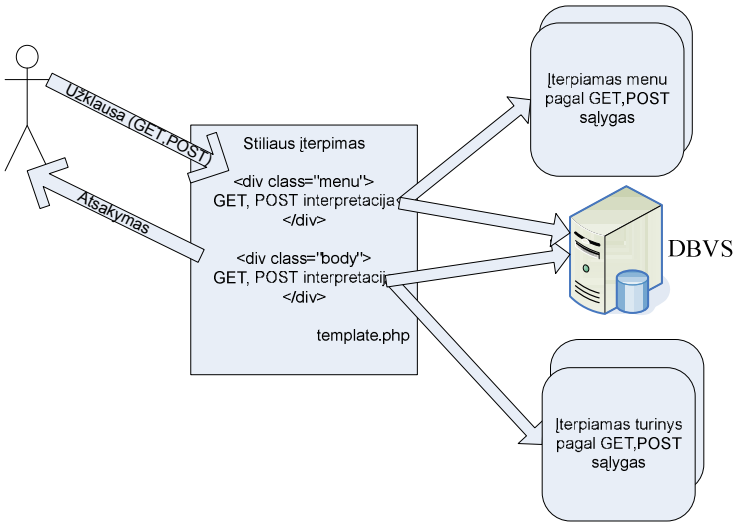
2.1.3 Įterpiamų puslapių minimizavimas

Dažnai žiniatinklio aplikacijos pagrindinis dokumentas yra sudarytas iš kelių įterpiamų puslapių, todėl dokumento struktūra yra modulinė. Kadangi yra keletas įterpiamų puslapių, kiekvieno įterpimas reikalauja papildomo laiko (5 pav.). Norint minimizuoti puslapių įterpimo laiką, reikia minimizuoti įterpiamų puslapių kiekį [7].



5 pav. Įterpiamų failų kiekio priklausomybė nuo laiko

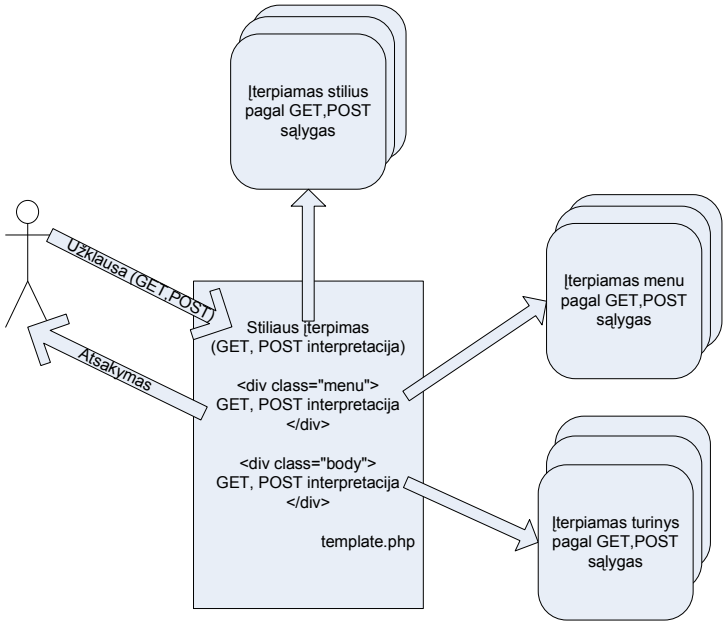
Didesnių spartos rezultatų galima pasiekti minimizavus failų įterpimus ir panaudojus duomenų bazių valdymo sistemas (6 pav.).



6 pav. Maksimalus failų įterpimų maksimizavimas. DBVS panaudojimas

2.1.4 Stiliaus pasirinkimas PHP kintamųjų pagalba

Stiliaus pasirinkimas atliekamas tokia pačia metodika kaip ir turinio įterpimas. Skirtumas tas, kad vietoje statinio stiliaus įkodavimo įterpiamas kodas dinamiškai parenkantis stiliaus failą pagal tam tikras sąlygas (7 pav.).



7 pav. Stiliaus, meniu ir turinio įterpimas pagal GET, POST sąlygas

2.1.5 Metodikos apibendrinimas

Tai pats lengviausias būdas suskirstyti kuriamą žiniatinklio puslapį į tam tikrus modulius ir lengvai atlikti tų modulių apjungimą, panaudojimą. Šios metodikos naudojimas nepašalina logikos ir dizaino atskyrimo problemos [8]. Todėl šios metodikos pavadinimas šablonų sistema būtų neteisingas. Pagrindiniai metodikos trūkumai yra negalėjimas atskirti

PHP ir HTML kodo, lėtas veikimas, netenkinimas minimalių šablonų sistemos reikalavimų. Apibendrinta savybių lentelė pateikiama 1 lentelėje.

1 lentelė. Standartinėmis priemonėmis paremtos šablonų metodologijos apibendrinimas

Savybės pavadinimas	Tinkamumo kriterijus	Komentaras
PHP kodo atskyrimas nuo HTML	Netenkinama	Neįmanoma atskirti HTML kodo nuo PHP, kadangi metodologija leidžia tik sukurti puslapio struktūrą. Galima turinį, meniu ir pan. perkelti į kitus failus.
Šablonų kompiliavimas	Netenkinama	Be PHP kodo atskyrimo nuo HTML savybės yra neįmanomas šablonų kompiliavimas.
Šablono papildinių (plugins) palaikymas	Netenkinama	Be PHP kodo atskyrimo nuo HTML savybės yra neįmanomas šablono papildinių palaikymas.
Šablono suderinamumas su XML	Netenkinama	Kadangi XHTML ir PHP maišoma kartu šis kodo dokumentas nesuderinamas su XML. Tačiau rezultatas gali būti teisingas XML dokumentas.
Modulinė struktūra	Tenkinama	Pagrindinis dokumentas sudaromas iš atskirų modulių.
Saugumas	Žemas	Kadangi PHP maišoma su HTML, puslapio dizaineris gali daryti įtaką PHP kodo veikimui.
Lengvumas naudotis	Gana painus	Nepatogus dizainerio ir koduotojo funkcijų atskyrimas.
Papildomų priemonių reikalingumas	Papildomų priemonių nereikia	Atliekama standartinėmis PHP programavimo kalbos priemonėmis.
Panaudojamumas	Tinkama smulkiems projektams	Dėl anksčiau išvardytų trūkumų, ši metodika netinkama stambiems ar vidutinio dydžio projektams.
Derinimas (*debugging*)	Papildomų priemonių nėra	Nėra palengvinančių priemonių klaidų paieškai ar ištaisymui.
Daugiakalbystės palaikymas	Nepalaikoma	Nėra priemonių, palengvinančių daugiakalbystės įgyvendinimą
Papildomų technologijų įsisavinimas	Papildomų technologijų įsisavinti nereikia	Kadangi ši metodika remiasi standartinėmis PHP priemonėmis, papildomų technologijų įsisavinti nereikia.

2.2 Paprastos šablonų sistemos

Paprastos šablonų sistemos yra programinė įranga, kuri apdoroja įeinantį tekstą (šabloną) ir pateikia išeinantį tekstą (atsakymą). Paprastų šablonų sistemų tikslas yra atskirti HTML kodą nuo PHP, jų nemaišyti kartu. Sudėtingos šablonų sistemos nuo paprastų sistemų skiriasi tuo, kad jų tikslas yra atskirti logiką nuo atvaizdavimo bei suteikti papildomų savybių: kešavimas, įskiepių palaikymas, suderinamumas su XML, saugumo maksimizavimas, greičio praradimo minimizavimas. Tai gali pasirodyti nedidelis skirtumas, bet tai yra labai svarbu. Galiausiai visų šablonų tikslas nėra pašalinti visą logiką iš HTML. Šablonų sistemos tikslas turi būti atskirti prezentacinę logiką nuo verslo logikos [9].

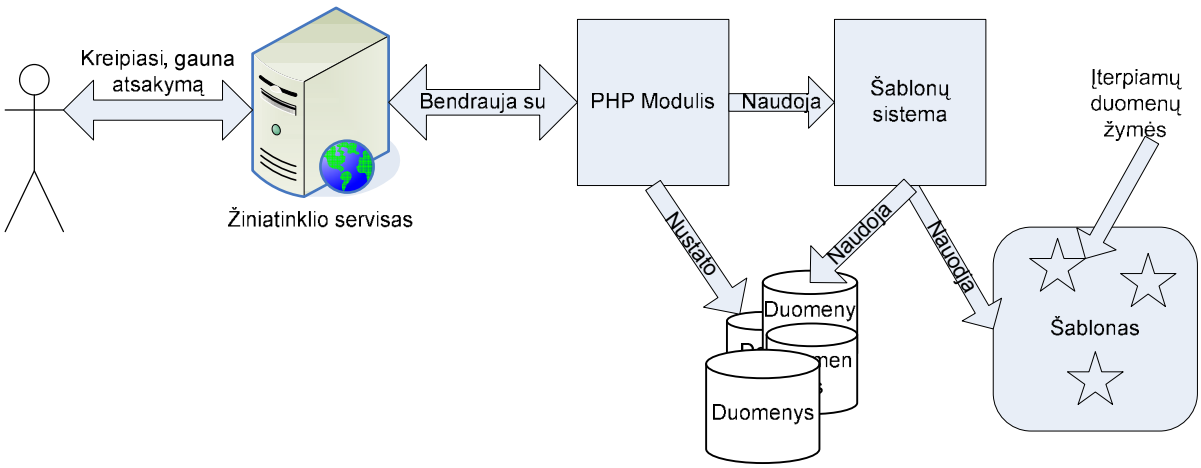
Populiariausios paprastos šablonų sistemos yra FastTemplate[10], Phemplate[11], bTemplate [12] ir daugelis kitų.

2.2.1 Paprastų šablonų sistemų apžvalga

Paprastos šablonų sistemos paprasčiausiai atskiria serverinės pusės kodą nuo klientinės pusės (PHP ir HTML atskyrimas). Šablonų sistemos išsprendžia šias dvi pagrindines problemas:

1. Kaip įgyvendinti prezentacinės logikos atskyrimą nuo verslo logikos;
2. Kaip atskirti sudėtingą PHP kodą nuo HTML.

Šios metodikos idėja yra įgalinti HTML puslapių dizainerius neturinčius PHP kodavimo patirties pakeisti puslapių dizainą, neatsižvelgiant į PHP kodą. PHP programuotojui turi būti sudaryta galimybė visą dėmesį skirti tik į kodavimą, nekreipiant dėmesio į dizaino elementus.



7 pav. Paprastos šablonų sistemos realizacija

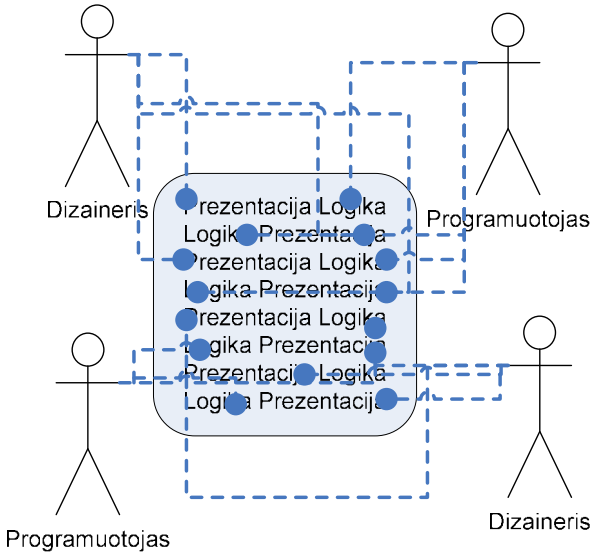
Šablonų sistemų atsiradimas taip pat įnešė papildomą sudėtingumo lygmenį žiniatinklio aplikacijų kūrimo procese. Pirmiausiai kelis kartus padidėjo projektą sudarančių failų skaičius. Dažniausiai vienas PHP failas yra atsakingas už verslo logiką, kitas už dizaino struktūrą bei vidinis turinio šablonas, kuris įstatomas į pagrindinį šabloną. Prie šių failų prisideda ir šablonų sistemos klasių, jų realizacijų failai.

Panaudojant šablonų sistemas skriptų vykdymas taip pat tampa sudėtingesnis. Šablono failai turi būti ne tik įterpiami, bet jie privalo būti gramatiškai teisingi. Taigi tai yra svarbus greičio praradimo momentas, paprastos šablonų sistemos neturi papildomų greitį spartinančių priemonių kaip kešavimas ar kompiliavimas.

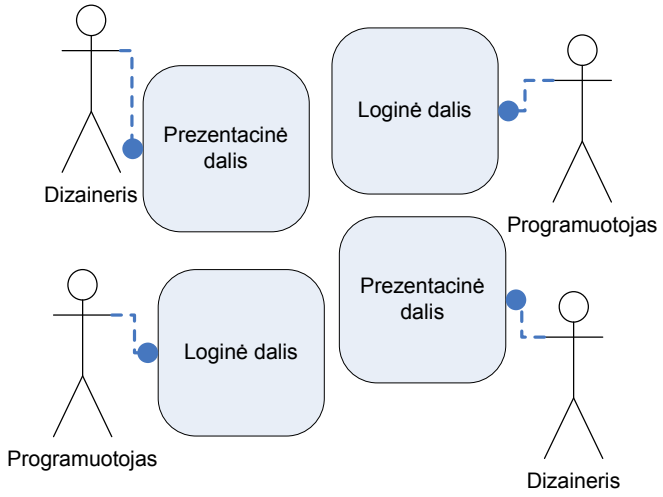
Šablonų sistema yra pseudokalba PHP skriptų interpretatoriui, kuri suteikia šablono kintamųjų interpoliavimą, ciklus ir kt. operacijas.

2.2.2 HTML dizainas ir aplikacijų programavimas

HTML dizaino darbai ir aplikacijų programavimas yra du skirtingi vaidmenys žiniatinklio aplikacijų kūrimo komandoje, dažnai šios užduotys atliekamos skirtingų žmonių bei skirtingu laiku [13]. Atskyrus HTML nuo logikos kodavimo, kiekvienas asmuo galės dirbti darbus atskirai, netrukdydami vienas kitam (9 pav.). Nenaudojant šablonų sistemų žiniatinklio aplikacijų kūrime, komandinis darbas yra sunkiai suderinamas arba visai neįmanomas (8 pav.).



8 pav. Komandinis žiniatinklio aplikacijos kūrimo darbas nenaudojant šablonų sistemų



9 pav. Komandinis žiniatinklio aplikacijos kūrimo darbas naudojant šablonų sistemų

2.2.3 Šablono žymėjimo stiliai

Šablono žymėjimas yra vienos pusės tiltas tarp šablono ir kodo. Žymėjimas atliekamas HTML šablone ir reprezentuoja vietas, kuriose šablonas galima manipuliuoti duomenimis arba kur bus įterpiami šablono duomenys.

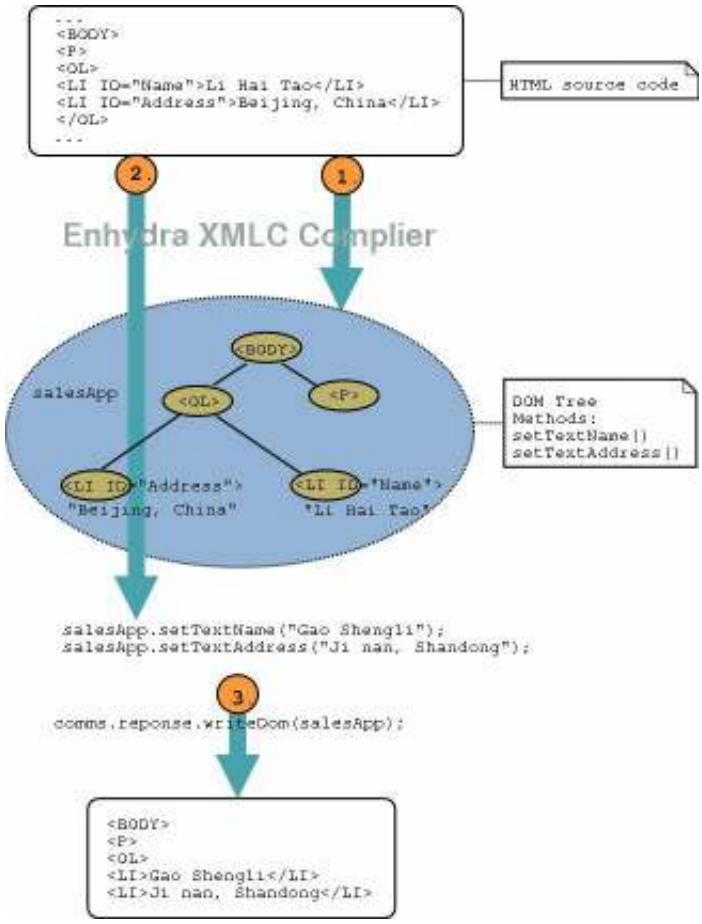
Skirtingos šablonų sistemos dažniausiai naudoja skirtingus žymėjimo stilius. Vieningo standarto lig šiol sukurta nėra.

2.2.3.1 Griežto HTML žymėjimo stilius (*Strict HTML*)

Šiame stiliuje visi žymekliai yra teisingi HTML operatoriai (*valid HTML*). Šablono žymėjimo pavyzdys pateiktas (10 pav.).

Privalumai: Šablonas yra teisingas XML ir XHTML dokumentas.

Trūkumai: Naujos sintaksės mokymasis.



10 pav. HTML atributais paremtas šablono žymėjimas

2.2.3.2 Specialios žymės (*Custom Tags*)

Programuotojo sukurtos žymės yra derinamos su HTML žymėmis (11 pav.).

```
<ERRORSUMMARY name='Summary'>
    <list:LISTITEM>
      <FONT COLOR="RED">{ErrorMessage}</FONT>
      <list:SEPARATOR><BR></list:SEPARATOR>
    </list:LISTITEM>
</ERRORSUMMARY>
```

11 pav. Specialios žymės derinamo su HTML žymėmis

Privalumai: Papildomo žymėjimo stiliaus panaudojimas.

Trūkumai: Painus ženklinimo atpažinimas.

2.2.3.3 HTML komentarų tipo žymės

Šis žymėjimo stilius kintamųjų atskyrimui naudoja HTML komentarų <!– Comment –> sintaksę (12 pav.).

```
<table>
  <!-- BEGIN_ROW -->
  <tr>
    <td><!-- Username --></td>
    <td><!-- Password --></td>
  </tr>
  <!-- END_ROW -->
</table>
```

12 pav. Šablono kintamiesiems atskirti naudojama HTML komentarų sintaksė

Privalumai: Aiški žymėjimo sintaksė.

Trūkumai: Sunkus suderinamumas su XML ir XHTML.

2.2.3.4 Šablonų sistemų kūrėjų parinkta žymėjimo sintaksė

Šis žymėjimo būdas įgyvendina tam tikrą šablono programavimo kalbą. Žymėjimas savo apibrėžta sintakse aprašo sąlygos sakinius, ciklus, kintamųjų modifikatorius ir pan. (13 pav.)

```
<table>
{section loop=users}
{strip}
  <tr bgcolor="{cycle values="aaaaaa,bbbbbb"}">
    <td>{users.name}</td>
    <td>{users.email}</td>
  </tr>
{/strip}
{/section}
</table>
```

13 pav. šablonų kūrėjų parinkta žymėjimo sintaksė

Privalumai: Griežtas verslo logikos ir atvaizdavimo atskyrimas

Trūkumai: Suderinamumo problemos su XML, naujos programavimo kalbos mokymasis.

2.2.3.5 Gimtoji PHP skriptų įterpimo žymėjimo sintaksė

Šis žymėjimo stilius naudoja PHP skripto įterpimo žymėjimo sintaksę. Kaip tik PHP kalba buvo kurta perteikti (atvaizduoti) C/C++ išvedamą kodą (14 pav.).

```
<tablo>
<?php
foreach ( $users as $user ) {
?>
  <tr>
    <td><?=$user->name()?></td>
```

```
    <td><?=$user->email()?></td>
  </tr>
<?php
}
?>
</tablo>
```

14 pav. Šablono žymėjimas panaudojat PHP sintaksę

Privalumai: Paprasta metodika. PHP sintaksė yra palaikoma daugelio žiniatinklio aplikacijų kūrimo įrankių ir derinimo programų (*debugger*).

Trūkumai: Puslapio dizaineris turi išmanyti PHP. PHP saugumo modelis šablone neleidžia naudotis visomis PHP galimybėmis. Šablonas tampa priklausomas nuo PHP programavimo kalbos.

2.2.4 Metodikos apibendrinimas

Naudojant paprastas šablonų sistemas lengvai galima atskirti žiniatinklio aplikacijos logiką nuo dizaino. Šios šablonų sistemos tinkamos mažo ir vidutinio dydžio projektams, kuriems nereikalingos įmantresnės šablonų savybės kaip spartumas, papildomų modulių palaikymas, suderinamumas su XML. Tai plačiausiai paplitusios šablonų sistemos tarp žiniatinklio aplikacijų kūrėjų. Detalus paprastų šablonų sistemų apibendrinimas pateiktas 2 lentelėje.

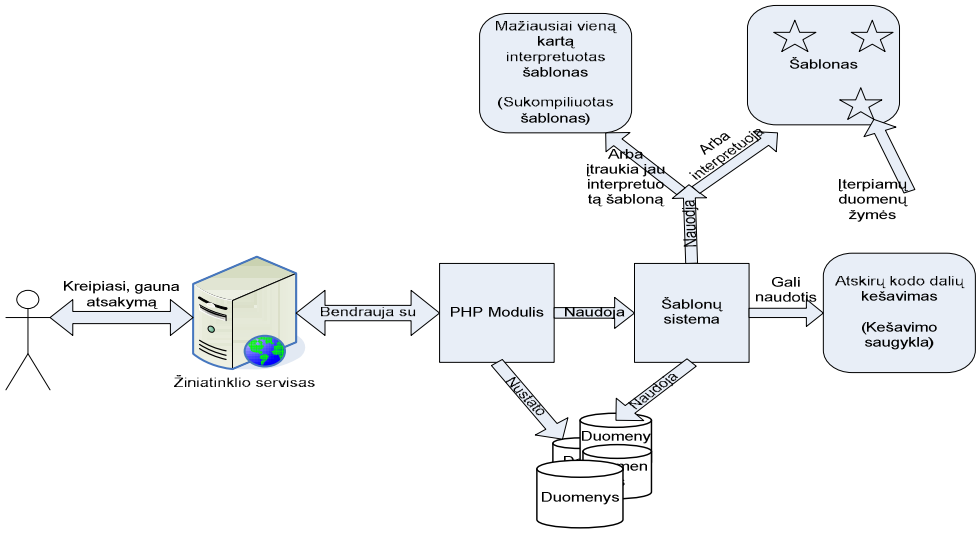
2 lentelė. Paprasčiausių šablonų sistemų apibendrinimas

Savybės pavadinimas	Tinkamumo kriterijus	Komentaras
PHP kodo atskyrimas nuo HTML	Tenkinama	Paprastuose šablonų sistemose gana griežtai tenkinamas PHP kodo atskyrimas nuo HTML, bet neskiriamas dėmesys verslo logikos atskyrimui nuo atvaizdavimo.
Šablonų kompiliavimas	Netenkinama	Paprastos šablonų sistemos dažniausiai neturi šios savybės.
Šablono papildinių (*plugins*) palaikymas	Netenkinama	Paprastos šablonų sistemos dažniausiai neturi šios savybės.
Šablono suderinamumas su XML	Nepakankamai	Daugelis paprastų šablonų sistemų nėra suderinamos su XML, kadangi naudoja savo apibrėžtas žymėjimo sintakses arba tenkina XML sąlygas su labai dideliais apribojimais.
Modulinė struktūra	Tenkinama	Web aplikacijų kūrimas panaudojant paprastas šablonų sistemas dažniausiai remiasi moduline struktūra.
Saugumas	Vidutinis	Didesnė pusė paprastų šablonų sistemų šablone riboja galimus panaudoti operatorius, funkcijas ar komandas, kurios gali įtakoti žiniatinklio aplikacijos saugumą. Šabloną dažniausiai kuria dizaineris nekompetentingas panaudoti PHP funkcijas, kurios gali įtakoti žiniatinklio aplikacijos saugumą.

Lengvumas naudotis	Vidutinis	Lengvumo naudotis savybę įvertinti vienareikšmiškai sunku.
Papildomų priemonių reikalingumas	Reikalingos paprastų šablonų sistemų klasės.	Reikia įdiegti papildomas klases, modulius norint naudotis paprastomis šablonų sistemomis. Standartinių priemonių neužtenka.
Panaudojamumas	Tinkama smulkiems ir vidutinio dydžio projektams	Ši metodika tinkama smulkiems ir vidutinio dydžio projektams. Netinkama dideliuose projektuose, kadangi yra ganėtinai lėta technologija, neturinti papildomų įskiepių panaudojimo galimybės, ribota savo funkcionalumu.
Derinimas (*debugging*)	Papildomų priemonių nėra	Nėra palengvinančių priemonių klaidų paieškai ar ištaisymui.
Daugiakalbystės palaikymas	Nepalaikoma	Kadangi nėra naudojami papildomi įskiepiai ar moduliai, sunku realizuoti daugiakalbystės reikalaujančiose projektuose.
Papildomų technologijų įsisavinimas	Reikalingas papildomas mokymasis	Programuotojams nėra įmanoma išvengti papildomo mokymosi. Efektyviai norint naudotis, reikia išmanyti paprastos šablonų sistemos klasių struktūrą, panaudojimo galimybes ir pan. Dizaineriams dažniausiai tenka įsisavinti šablono žymėjimo metodiką.

2.3 Šablonų sistemos, palaikančios kompiliavimą ir papildomus modulius

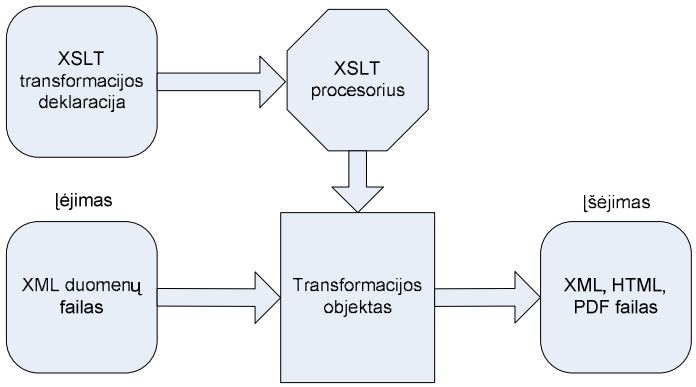
Dideliuose projektuose paprastų šablonų sistemų panaudojimas yra mažai įmanomas. Dideli projektai susideda iš tūkstančių failų ir visų tų failų integravimas į vieną visumą iškelia greičio ir saugumo problemas. Norint maksimaliai sumažinti šablonų interpretavimo laiką buvo sugalvotas šablonų kompiliavimas ir kešavimas. Sudėtingų šablonų ideologija remiasi paprastų šablonų ideologija, tik žymiai sugriežtina idėja, kad ne tik HTML turi būti atskirtas nuo PHP, bet ir verslo logika turi būti nepriklausoma nuo atvaizdavimo [14]. Sudėtingų šablonų sistemų schema pateikiama (15 pav.).



15 pav. Sudėtingos šablonų sistemos prototipas

2.4 XSLT transformacijomis paremta šablonų sistema

XSL transformacijos arba XSLT yra XML pagrindu sukurta kalba, naudojama transformuoti XML dokumentus [15]. Transformuojamas XML dokumentas nėra keičiamas, jis yra tik įėjimas XSLT transformacijų sistemai, pagal šio dokumento turinį yra sukuriamas naujas transformuotas XML dokumentas (16 pav.). Naujas XML dokumentas apdorotas XSLT procesoriaus gali būti išvedamas standartine XML sintakse arba kitais populiariais formatais: HTML, tiesiog tekstu ar pdf dokumentu [16]. Dažniausiai XSLT yra naudojamas norint transformuoti XML duomenų failą į Web puslapius ar PDF dokumentus. XML duomenų failo transformavimas į žiniatinklio puslapius kaip tik yra viena iš šablonų sistemos realizacijų. Transformacijomis paremta šablonų sistema nuo anksčiau aptartų šablonų skiriasi tuo, kad šablone pažymėtų žymių įstatymo kintamaisiais požiūris pakeičiamas į XML duomenų failo transformavimą į norimą duomenų atvaizdavimo formą (*HTML, PDF*).



16 pav. XSLT modelis

Vienas iš XSLT privalumų yra galimybė apsirašyti kelias transformacijas (atvaizdavimus) vienam duomenų failui. Turint vieną XML failą galima lengvai sukurti skirtingai atrodančius žiniatinklio puslapius.

XSLT transformacijos gali būti apdorojamos klientinėje pusėje, t.y. naršyklės pagalba, bet žymiai patogiau XSLT transformacijas integruoti į žiniatinklio aplikaciją ir jas atlikti serverio pusėje. Kliento pusėje transformavimas gali įšaukti nemažai problemų, tokių kaip naršyklės nesuderinamumas su XSLT, netinkamas interpretavimas ir pan.

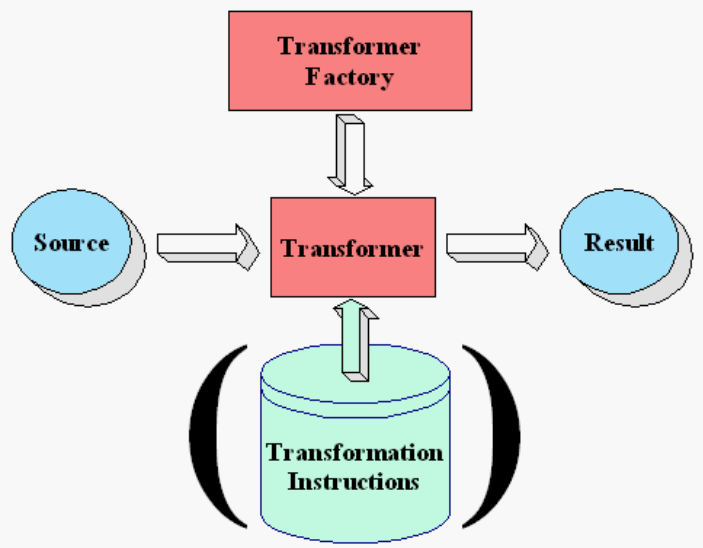
2.4.1 Transformacijų metodologijos

XSLT kūrėjų rekomendacijose nepateikiami patarimai kaip transformacijos turi būti įgyvendinamos, aprašomas tik implementuotos XSLT sistemos veikimas [17]. XSLT transformacijų sistemos (variklio) pasirinkimas priklauso nuo naudojamos programavimo kalbos ar sistemos. Dažniausiai kiekviena programavimo kalba, sistema pateikia atskirą bibliotekų rinkinį skirtą XSLT įgyvendinimui.

2.4.1.1 XML transformacijų API (*TrAX*)

Šiuo metu vienas populiariausių būdų atlikti transformacijas yra transformacijų API skirtų XML dokumentų transformavimui (*TrAX*) panaudojimas (17 pav.). TrAX metodologija buvo kurta Java programuotojų auditorijai. Norint įvertinti metodologijos naudą užtenka vieno fakto, tai kad ši metodika buvo įgyvendina daugelyje programavimo kalbų. TrAX panaudojimas apima tris pagrindinius žingsnius [18]:

1. TransformerFactory klasės objekto sukūrimas.
2. Pasinaudojant sukurtu TransformerFactory klasės objektu ir konstruktoriuje nurodoma XSLT transformacijos deklaracija (*XSLT stylesheet*) yra sukuriamas Transformer klasės objektas. Šis objektas reikalingas transformacijų atlikimui, vėliau visi transformuojami XML failai naudosis šia XSLT transformacijos deklaracija, kurios failo pavadinimas buvo pateiktas kaip konstruktoriaus parametras.
3. Transformer klasės objektas turi iškviesti transform() metodą, kuriame nurodomas XML įėjimas ir atliktos transformacijos rezultato objektas.



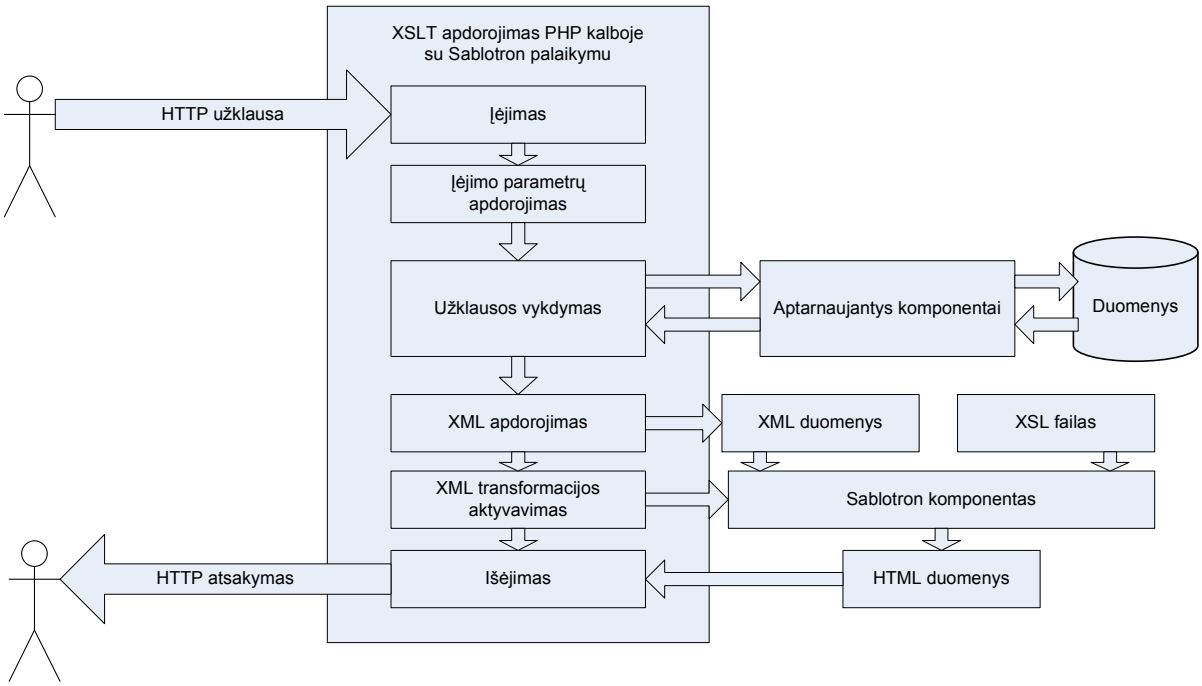
17 pav. TrAX modelis

2.4.1.2 PHP ir transformacijų sistema Sablotron

Sablotron XSLT procesorius yra kompanijos Ginger Alliance LTD. produktas, neseniai įtrauktas į didžiulę PHP plėtinių (*extensions*) saugyklą. XSLT panaudojimas PHP kalboje su Sablotron palaikymu yra nesudėtingas ir remiasi ankščiau pateikta TrAX metodologija. Reikia pažymėti, kad daug TrAX metodologijos savybių yra nepalaikomos ar neįgyvendintos. Kompiliavimas PHP su Sablotron palaikymu nėra labai lengvas procesas. Prieš kompiliuojant reikia tiek PHP, tiek Sablotron programiniam kodui pritaikyti atnaujinimus (*patch*) bei PHP sukonfigūruoti su XSLT palaikymu.

XML/XSL transformacijos proceso modelis panaudojant Sablotron transformacijų procesorių pateiktas (18 pav.). Transformacijos vykdymo eiliškumas yra toks [19]:

1. Gaunama HTTP užklausa, t.y. įėjimas į Web aplikaciją;
2. Įėjimo informacija yra apdorojama ir sukuriama duomenų struktūra;
3. Užklausa yra vykdoma. Tai gali reikšti operacijos aktyvavimą viename iš aptarnaujančių komponentų, pavyzdžiui, informacijos skaitymas ar rašymas į duomenų bazę. Gali būti gražinami duomenys ir / arba klaidų pranešimai;
4. Apdorotas komponento rezultatas yra išsaugomas į XML dokumentą;
5. XML transformacijos aktyvavimo komponentas iškviečia Sablotron transformacijų sistemą;
6. Sablotron transformacijų komponentas XML failą transformuoja į HTML ir jį perduoda išėjimo komponentui. Galiausiai XML transformacija pateikiama užklausą suformavusiam agentui.



18 pav. XML/XSL transformacijos procesas panaudojant Sablotron

Dinaminių Web puslapių kūrimas, pasinaudojant XML ir XSLT, įgauna vis didesnį populiarumą visuose interneto bendruomenėse. Pagrindiniai privalumai:

- XML ir XSL yra atviri standartai, palaikomi W3 konsorciumo;
- Verslo logikos atskyrimas nuo atvaizdavimo;
- XHTML ir WML yra XML pagrindo, kaip tik XSLT natūrali prigimtis siejasi su XML. Todėl garantuojamas visapusiškas suderinamumas.

Metodologija turi ir nemažai trūkumų. Esminiai trūkumai yra sudėtingas technologinis sprendimas, naujos metodikos įsisavinimas, ganėtinai lėtas sprendimas lyginant su kompiliuotais šablonais.

2.4.2 Metodikos apibendrinimas

XSLT yra atskiras standartas, kurio tikslas yra XML dokumentą transformuoti į kitą formatą XML, XHTML, PDF. Kadangi XSLT yra standartas, o ne papildomas klasių rinkinys vienai ar kitai programavimo kalbai, pasinaudoti šia metodika galima beveik visose programavimo sistemose. Projektai, kuriuose duomenų manai atliekami XML pagalba, XSLT yra pats patogiausias būdas šiuos XML dokumentus transformuoti į XHTML ar PDF.

Viena problematiškiausių vietų panaudoti XSLT transformacijas PHP kalboje – suderinamumo stoka. Norint įdiegti XSLT į PHP reikalaujama papildomų bibliotekų įdiegimo bei PHP perkompiliavimo. Dažnai realizuotas projektas įgyvendinantis XSLT transformacijas būna lėtesnis už šablonų sistemas su kompiliavimo palaikymu. Išlieka komplikuota dizainerio darbo specifika. Žiniatinklio puslapių dizaineris turi gerai išmanyti XHTML, XSL, WML. Naudojant šią metodiką žiniatinklio aplikacijų kūrime labai išauga reikalavimai dizainerių klasifikacijai, tai neigiamai veikia projekto biudžetą.

Detalus XSLT transformacijomis paremtų šablonų sistemų apibendrinimas pateiktas 3 lentelėje.

3 lentelė. XSLT transformacijomis paremtų šablonų sistemų apibendrinimas

Savybės pavadinimas	Tinkamumo kriterijus	Komentaras
PHP kodo atskyrimas nuo HTML	Tenkinama	Griežtas PHP kodo atskirimas nuo HTML. PHP komponentas atsakingas tik už verslo logiką ir transformacijos atlikimą. Dizaineris aprašo kaip transformuoti PHP suformuotą XML dokumentą.
Šablonų kompiliavimas	Netenkinama	XSLT kompiliavimas nepalaikomas.
Šablono papildinių (*plugins*) palaikymas	Netenkinama	Papildomi moduliai į XSLT neįstatomi.
Šablono suderinamumas su XML	Visapusiškai suderinama su XML	XSLT pagrindas yra XML.
Modulinė struktūra	Dalinai	Už modulės struktūros sukūrimą didžiąja dalimi atsakingas sistemos kūrėjas. Metodika padeda suskaldyti kuriamą žiniatinklio aplikaciją į šias dalis: • Logiką; • Duomenis; • Atvaizdavimo aprašymą.
Saugumas	Vidutinis	XSLT transformacija gali būti vykdoma tiek serverio, tiek klientinėje (panaudojus interneto naršyklę) pusėje. Klientinėje pusėje vykdoma XSLT transformacija gali sukelti saugumo problemų.

Lengvumas naudotis	Sudėtingas	Sudėtingas dizainerio darbas. Didžiausia problema - XSLT transformacijų kalbos įsisavinimas.
Papildomų priemonių reikalingumas	Reikia PHP sukompiliuoti su XSLT palaikymu	Gana sudėtingas XSLT įjungimas į PHP. Reikalingas papildomų bibliotekų įdiegimas ir PHP perkompiliavimas.
Panaudojamumas	Tinkama vidutinio dydžio ir dideliems projektams	Kadangi metodika yra gana sudėtinga, ji pasiteisina tik didesnės apimties projektuose.
Derinimas (*debugging*)	Papildomų priemonių nėra	Papildomos priemonės klaidų pašalinimui nepateikiamos.
Daugiakalbystės palaikymas	Lengva atlikti standartinėmis priemonėmis	Daugiakalbystės palaikymas atliekamas transformacijų sistemai pateiktus kitos kalbos XML dokumentą. Papildomas kodavimas nereikalingas.
Papildomų technologijų įsisavinimas	Reikalingas papildomas mokymasis	Reikalingas panaudojimo mokymasis, XSLT transformacijų kalbos įsisavinimas. Žiniatinklio dizaineriui nedirbusiam su programavimo užduotimis pereiti prie šios metodikos gali būti neįmanoma užduotis.

2.5 Modelis – Vaizdas – Valdiklis struktūrinės šablonų sistemos

Modelis – Vaizdas – Valdiklis (*MVC*) yra programinės įrangos architektūra, kuri aprašo kaip suskaidyti aplikacijos duomenų modelį, vartotojo sąsają ir kontrolės logiką į tris skirtingus komponentus [20]. Vieno komponento modifikavimas turi minimaliai įtakoti kitus komponentus, idealiausiai jų visiškai neįtakoti.

MVC dažnai yra vadinamas programinės įrangos kūrimo šablonas. Tačiau MVC daugiau apima architektūrinę dalį negu tipinis kūrimo šablono apibūdinimas. Teisingiausias MVC terminas yra pasiūlytas Buschmann architektūrinis šablonas arba agregatinis kūrimo šablonas [21].

2.5.1 Modelis

Modelis yra specifinės srities informacijos reprezentacija, funkcionalumo šerdis, kurioje programinė įranga veikia. MVC tikslas yra atskirti modelį nuo vaizdo ir valdiklio, kurie kartu suformuoja programinės įrangos grafinę sąsają.

Modelis reprezentuoja verslo duomenis ir taisykles, kurios valdo priėjimą ir keitimą šių duomenų. Dažniausiai modelis panaudojamas kaip programinės įrangos aproksimacija realaus gyvenimo procesui. Realaus gyvenimo modeliavimo technikos pritaikomos apibrėžiant modelį.

Jokių specifinių žinių modelis apie vaizdus ar valdiklius neturi. Sistema pati prižiūri sąryšius tarp modelio ir vaizdo bei praneša vaizdui kada modelio būsena pasikeičia. Vaizdas

valdo duomenų, kuriuos reprezentuoja modelis, atvaizdavimą. Modelis gali turėti daugiau negu vieną vaizdą.

2.5.2 Vaizdas

Vaizdas yra skirtas modelio suformuotų duomenų atvaizdavimui galutiniam vartotojui. Vaizdas reprezentuoja išėjimą iš programinės įrangos.

Vaizdas turi laisvą priėjimą prie modelio, bet jam draudžiama pakeisti modelio būseną. Vaizdai yra tik skaitymo modelio būsenos reprezentacija. Duomenų skaitymas iš modelio atliekamas standartiniais modelyje aprašytais metodais.

2.5.3 Valdiklis

Valdiklis priima ir transliuoja įėjimą į užklausas, kurios skirtos modeliui arba vaizdui.

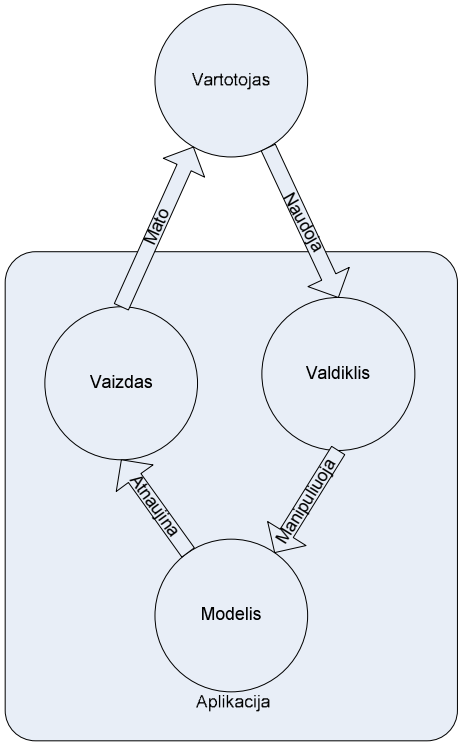
Paprastai kontroleris yra atsakingas už modelyje esančių metodų iškvietimą, kurie pakeičia iškviečiamo modelio būseną. Interpretuojamos pelės arba klaviatūros įvestys iš varotojo, pagal jas yra atliekami modelyje arba / ir vaizde atitinkami pakeitimai.

Valdiklio pagalba vartotojas sąveikauja su aplikacija. Valdiklis priima vartotojo įvestį ir duoda nurodymus modeliui ir vaizdui įvykdyti veiksmus pagal pateiktą įvestį.

Valdiklis transliuoja sąveiką su vaizdu į veiksmus, kuriuos turi įvykdyti modelis. Pavyzdžiui, žiniatinklio aplikacijose ši sąveika atliekama HTTP GET ir POST užklausomis. Veiksmai, kuriuos atlieka modelis, apima verslo procesų aktyvavimą ar modelio būsenos keitimą. Pagal vartotojo įvestį ir modelio suformuotą rezultatą, valdiklis parenka atitinkamą vaizdą.

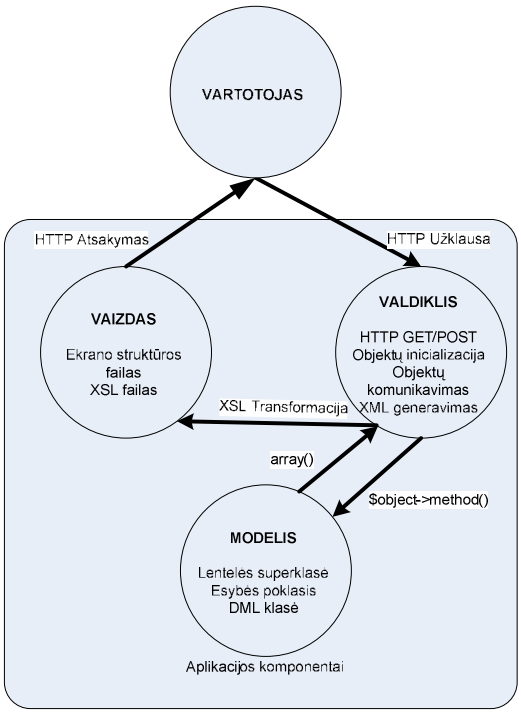
2.5.4 Komponentų tarpusavio sąryšiai

Modelis, vaizdas ir valdiklis yra artimai vienas su kitu susiję ir nuolat tarpusavyje kontaktuojantys. (19 pav.) pateikta MVC paradigmos ryšių iliustracija [22]. Gana sunku yra pasiekti griežtą modelio ir vaizdo atskyrimą.



19 pav. Pagrindiniai MVC sąveikos komponentai

Žiniatinklio aplikacijos realizacija panaudojant MVC ir XSLT pateikiama (20 pav.).



20 pav. Žiniatinklio aplikacijos realizacija naudojant MVC paradigmą ir XSLT transformacijų sistemą

2.5.5 MVC paradigmos privalumai

2.5.5.1 Lengvai keičiama vartotojo sąsaja

Skirtingi vaizdai ir valdikliai gali būti pakeičiami, norint sukurti aplikaciją su skirtingomis varotojo sąsajomis tam pačiam modeliui. Pavyzdžiui, tuo pačio modelio

duomenys gali būti parodomi ekrane kaip stulpelinė ar skirtulio formos diagrama, arba lentelės pavidalu.

2.5.5.2 Vienas modelis ir keli vaizdai tuo pačiu laiku

Keletas skirtingų vaidų gali būti aktyvuojami vienu metu. Kiekvienas vaizdas tuo pačiu metu ir nepriklausomai nuo kitų gali atvaizduoti tą pačią modelio informaciją. Šis savybė labiau tinkama grafinių aplikacijų kūrime panaudojant MVC paradigmą negu žiniatinklio aplikacijose.

2.5.5.3 Vaizdų sinchronizacija

MVC pateikia patogų pakeitimų propagavimo mechanizmą. Jei modelio būsena yra pakeičiama, keitimo propagavimo mechanizmas garantuoja, kad visi vaizdai vienu metu gaus naujausią modelio informaciją.

2.5.5.4 Lengvesnis testavimas

Panaudojant MVC projekto testavimas gali būti žymiai lengvesnis, kadangi aplikacijos funkcionalumo šerdis yra modulis. Norint ištestuoti funkcionalumą, reikės ištestuoti tik modelį.

2.5.6 MVC paradigmos trūkumai

2.5.6.1 Padidėjąs sudėtingumas

Norint naudotis MVC paradigma yra būtiną išmokti sudėtingus MVC programavimo karkasus (*Framework*), kurie skirtingose programavimo kalbose yra skirtingi. Kad būtų tenkinama MVC ideologija, būtina laikytis griežtų aplikacijų kūrimo taisyklių. Dažnai priimamų sprendimų negalima įgyvendinti vienareikšmiškai, būtina taikytis prie MVC paradigmos.

2.5.6.2 Artima vaizdų ir valdiklių sąsaja su modeliu

Pasikeitus modelio būsenai yra reikalingi lygiagretūs pakeitimai vaizde ir galimai valdiklyje. Tam tikri pakeitimai gali būti laibai sudėtingi.

2.5.6.3 Keitimo pranešimų perpildymas

Pakeitimų propagavimo mechanizmas gali būti neefektyvus, kai dažni modelio pakeitimai reikalauja daugelio keitimų pranešimų perdavimo.

2.5.6.4 Vaizdo ir valdiklio sugretinimas

Griežtas vaizdo atskyrimas nuo valdiklio yra neįmanomas arba sunkiai įgyvendinamas.

2.5.7 Metodikos apibendrinimas

Tai, kad reikia atskirti srities logiką nuo prezentacijos yra beveik neginčytina objektiškai orientuoto projektavimo aksioma. Lig šiol daugelis sėkmingų projektų šiuos metodikos netaiko. Standartinės PHP programavimo priemonės laisvai leidžia tarpusavyje maišyti vartotojo sąsajos ir srities logikos elementus. Vienas stambiausių programinės įrangos kūrėjų Microsoft .NET architektūroje netgi ragina srities logiką maišyti su vartotojo sąsaja.

Raginimas nenaudoti MVC paradigmos grindžiamas sudėtingumu ir sunkumu naudotis. Praktikoje griežtas trijų MVC pagrindinių komponentų atskyrimas yra sunkiai pasiekiamas arba neįmanomas.

Dažniausiai rekomenduojama aplikaciją sudalinti į komponentus atsakingus už įėjimą, vykdymą ir išėjimą. Jei pasirinktą užduotį lengviau įgyvendinti nusižengiant MVC paradigmai, tai geriausiai taip ir daryti. Griežtas MVC paradigmos laikymasis gali sužlugdyti projektą dėl jo sudėtingumo.MVC paradigmos apibendrinimas pateiktas 4 lentelėje.

4 lentelė. MVC paradigmos apibendrinimas

Savybės pavadinimas	Tinkamumo kriterijus	Komentaras
PHP kodo atskyrimas nuo HTML	Tenkinama	MVC paradigmos tikslas atskirti ne PHP kodą nuo HTML, bet verslo (srities) logiką atskirti nuo atvaizdavimo.
Šablonų kompiliavimas	Dažniausiai tenkinama	MVC dažniausiai palaiko šablonų kompiliavimą
Šablono papildinių (*plugins*) palaikymas	Dalinai	Pateikiama ribotas kiekis papildinių (*plugins*).
Šablono suderinamumas su XML	Dalinai	Šablonas yra dažniausiai nesuderinamas su XML, bet aplikacijos konfigūracija gali būti aprašyta XML.
Modulinė struktūra	Griežtai tenkinama	MVC aprašo šitokią aplikacijos struktūrą: • Modelis; • Vaizdas; • Valdiklis.
Saugumas	Aukštas	Aplikacijos logika yra atskiriama nuo atvaizdavimo, todėl nėra lengva tiesiogiai prieiti prie srities logikos aprašymo. Dažniausiai šablonuose nėra leidžiamas PHP kodo vykdymas.
Lengvumas naudotis	Labai sudėtingas	Didžiausia MVC paradigmos neigiama savybė yra sudėtingumas.
Papildomų priemonių reikalingumas	Reikalinga įdiegti papildomas struktūrines klases (*Framework*)	Struktūrinių klasių įdiegimas dažniausiai nėra sudėtingas. Trūkumas, kad PHP kalbai skirti yra tik keli MVC projektai.
Panaudojamumas	Tinkama vidutinio dydžio ir	Kadangi metodika yra gana

	dideliems projektams	sudėtinga, ji pasiteisina tik didesnės apimties projektuose.
Derinimas (*debugging*)	Papildomų priemonių nėra	Papildomos priemonės klaidų pašalinimui nepateikiamos.
Daugiakalbystės palaikymas	Netenkinama	Pateikiamos papildomos priemonės.
Papildomų technologijų įsisavinimas	Reikalingas papildomas mokymasis	Reikalingas panaudojimo mokymasis. Web dizaineriui nedirbusiam su programavimo užduotimis pereiti prie šios metodikos gali būti sunku.

2.6 Šablonų sistemų vertinimas ir išvados

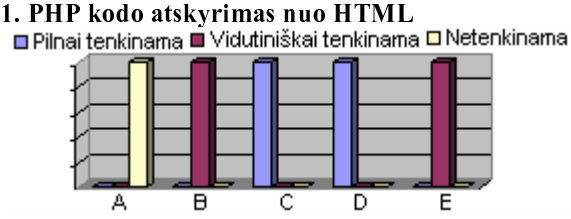
Norint įvertinti šablonų sistemas pagal įvestą taksonomiją, pirmiausiai reikia nustatyti vertinimo metrikas. Vertinant nebus prisirišta prie konkrečios šablonų sistemos realizacijos, projekto, bus vertinama tik pagal išskirtas šablonų grupes (Standartinėmis priemonėmis paremtas šablonų įgyvendinimas, Paprastos šablonų sistemos, Šablonų sistemos palaikančios kompiliavimą ir papildomus modulius, XSLT transformacijomis paremta šablonų sistema, Modelis – Vaizdas – Valdiklis struktūrinės šablonų sistemos).

5 lentelėje pateikiamas šablonų sistemų vertinimą.

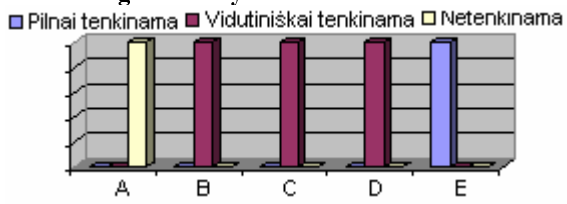
5 lentelė. Šablonų sistemų vertinimas

Žymėjimai:
A - Standartinėmis priemonėmis paremtas šablonų įgyvendinimas
B - Paprastos šablonų sistemos
C - Šablonų sistemos palaikančios kompiliavimą ir papildomus modulius
D - XSLT transformacijomis paremta šablonų sistema
E - Modelis – Vaizdas – Valdiklis struktūrinės šablonų sistemos

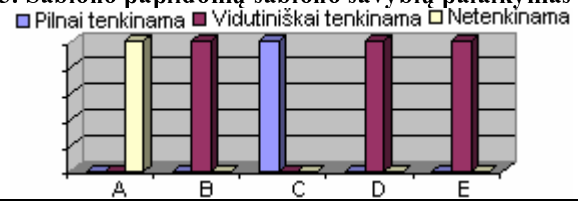
1. PHP kodo atskyrimas nuo HTML



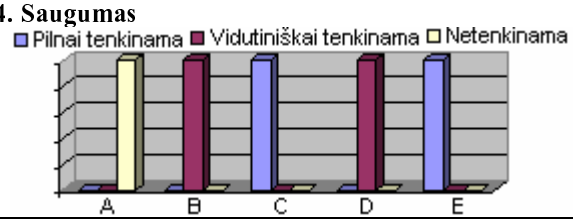
2. Srities logikos atskyrimas nuo atvaizdavimo



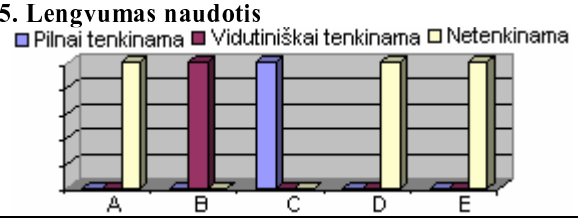
3. Šablono papildomų šablono savybių palaikymas



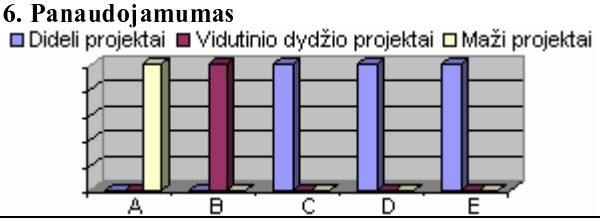
4. Saugumas



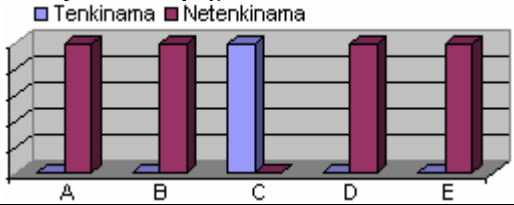
5. Lengvumas naudotis



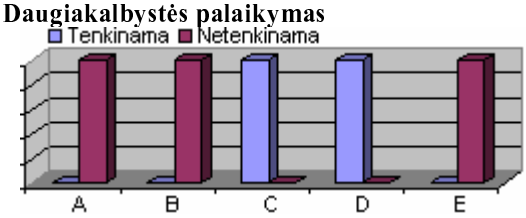
6. Panaudojamumas

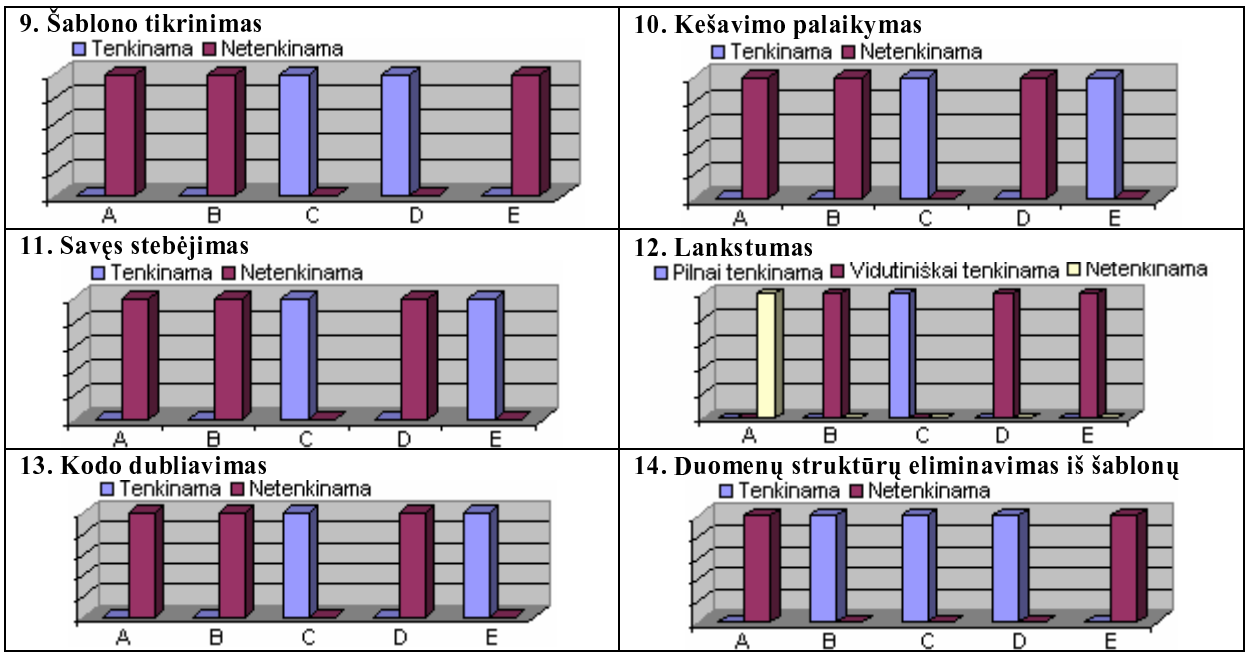


7. Klaidų šalinimo pagalba

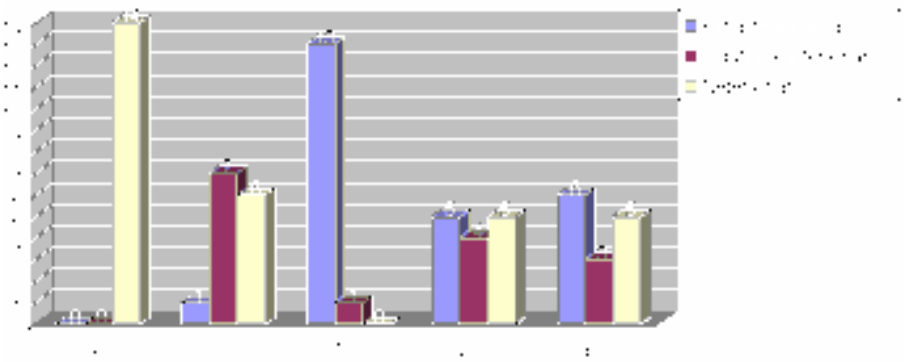


8. Daugiakalbystės palaikymas





Apibendrinus šablonų sistemų vertinimus galima išskirti šablonų sistemų grupes kurios labiausiai tenkina išskirtas metrikas (21 pav.).



21 pav. Šablonų sistemų apibendrintas vertinimas

3 Žiniatinklio aplikacijų kūrimo modelis

3.1 Žiniatinklio aplikacijų kūrimo modelio apžvalga

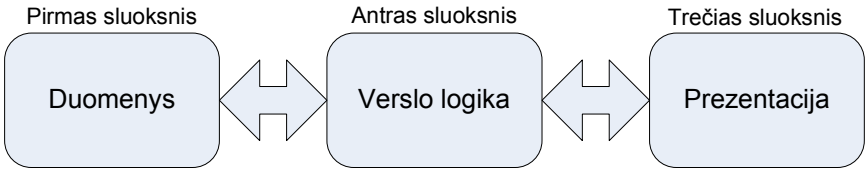
Verslo logikos ir prezentacijos atskyrimo realizavimas sudėtingų žiniatinklio aplikacijų kūrime yra vienas labiausiai siektinų tikslų. Daugelį atvejų, PHP panaudojimas yra kompromituojantis. Vieni kodo poaibiai įgyvendina savybes įterpiant atvaizdavimo žymenis verslo logikoje. Kiti realizuoja verslo logiką atvaizdavime.

Prezentacijos ir verslo logikos atskyrimo problema yra aktuali ir šiandienos projektuose. Mes pateiksime modelį kaip kurti žiniatinklio aplikacijas, jų kūrimą suskirstant į modulius atsakingus už prezentaciją, verslo taisykles ir duomenis. Realizaciją atliksime pasinaudodami PEAR::DB duomenų abstrakcijos klase bei viena galingiausia PHP kalbai sukurtų šablonų sistemų – Smarty.

Duomenų bazių abstrakcijų paketų rinkiniai yra įprasti daugelyje programavimo kalbų. Java programavimo kalboje įgyvendintas JDBC, Perl yra realizuota DBI duomenų abstrakcijos sąsaja. PHP kalba nėra išimtis yra keliatas duomenų bazių abstrakcijos klasių skriptų PHP. Aprašydami modelį panaudosime vieną iš naujausių ir plačiausiai paplitusių – PEAR::DB. Internetinių aplikacijų kūrime šablonų sistemų panaudojimas yra taip pat įprastas. Pasirinkta Smarty šablonų sistema yra kitokia negu egzistuojantys sprendimai, dėl savo paprastumo ir palaikomų savybių, tokių kaip šablonų kešavimas, kompiliavimas, kurių dėka kodo vykdymo laikas yra minimalus.

3.2 Trijų sluoksnių žiniatinklio kūrimo modelis

Aplikacijos projektavimas skirtingos paskirties sluoksniais yra naudingas dėl daugelio priežasčių. Kiekvienas loginis žiniatinklio aplikacijos sluoksnis gali būti vykdomas skirtingose kompiuteriuose, tai sąlygoje trumpesnį aplikacijos vykdomo laiką. Efektyvus sudalinimas į sluoksnius gali sukurti aplikacijos struktūrą, palengvina aplikacijos kodo palaikomumą, reikalavimų kaita yra lengviau įveikiama.

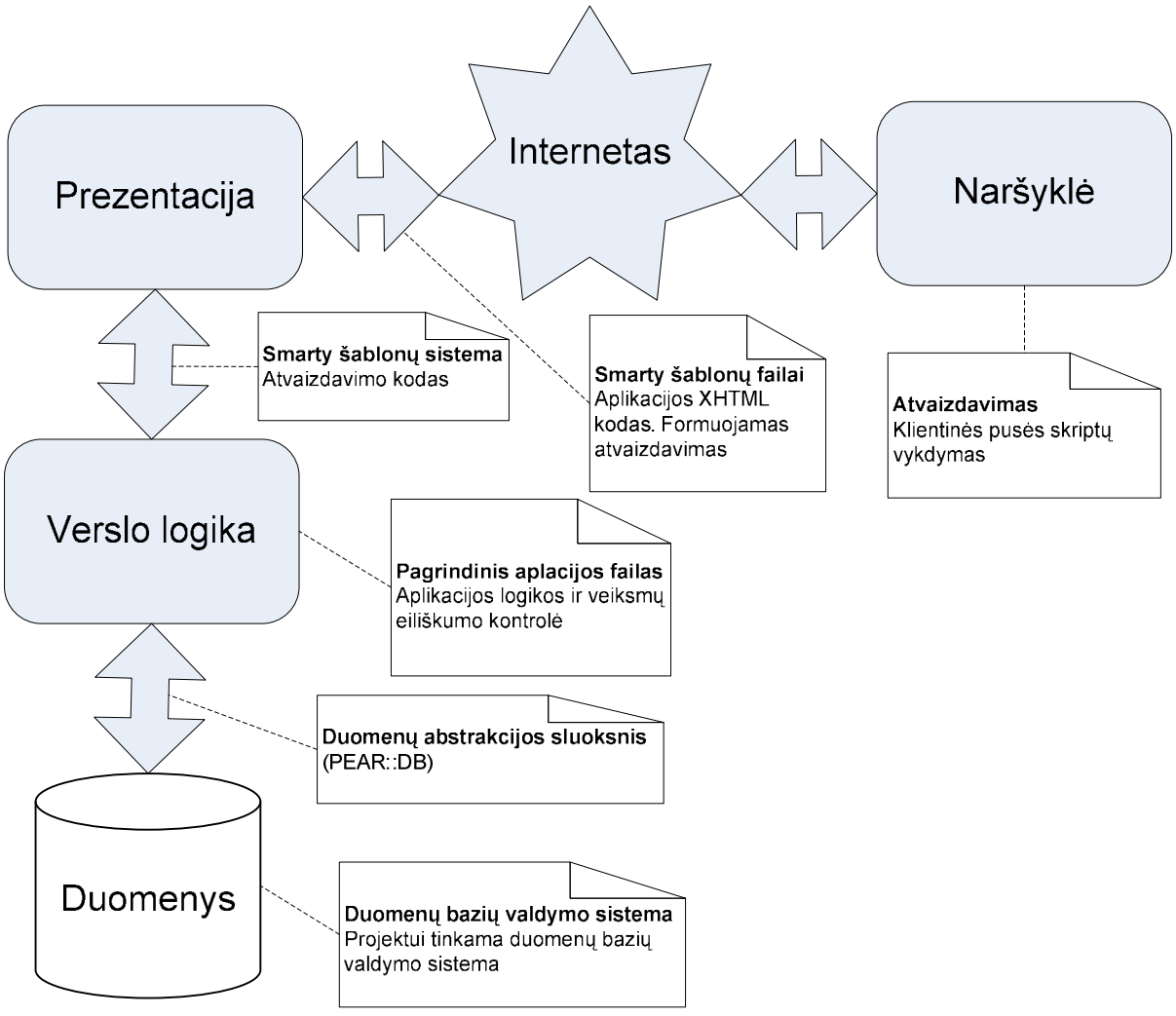


22 pav. Žiniatinklio aplikacijos pagrindinai trys sluoksniai

Trijų sluoksnių architektūroje, pirmas sluoksnis yra skirtas duomenų saugojimui. Pagrindinė technologija naudojama šios dalies įgyvendinimui gali būti kliento pusėje esantys „sausainiukai“, sesijos, serveryje talpinami duomenų failai arba duomenų bazių valdymo

sistemos. Dažnai šiuos technologijos derinamos tarpusavyje viena su kita. Atliekant projektą yra naudojamos sesijos ir duomenų bazių valdymo sistemos arba kliento pusėje esantys „sausainiukai“ ir duomenų bazių valdymo sistemos. Sekantis sluoksnis yra atsakingas už verslo logikos apdorojimą. Mes siūlome šiame sluoksnyje apdoroti pirmo sluoksnio duomenis, kurti taisykles įgalinančias manipuliuoti šiais duomenimis. Tai yra sluoksnis apimantis didžiąją žiniatinklio aplikacijos dalį. Paskutinis sluoksnis yra atsakingas už prezentaciją, tai - rezultatų atvaizdavimas, kurie yra sukuriami verslo logikos sluoksnyje.

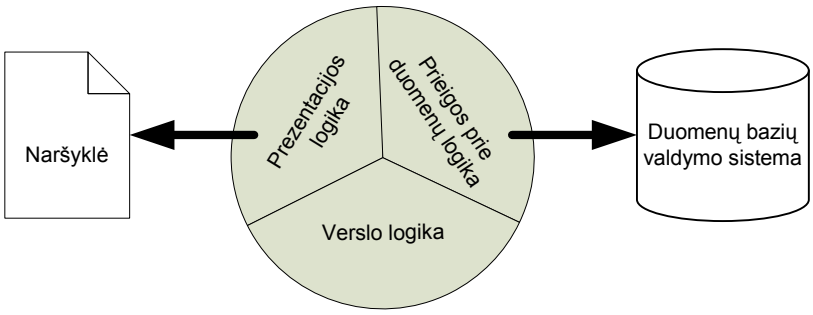
Mūsų siūlomo modelio grafinis atvaizdavimas pateiktas (23 pav.). Jame pateikiame sluoksnių priklausomybę bei kiekvieno sluoksnio technologinė realizacija.



23 pav. Siūlomas žiniatinklio aplikacijų kūrimo modelis

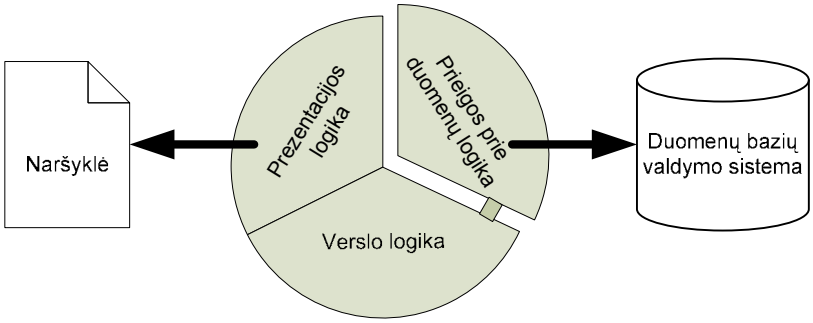
3.3 Trijų sluoksnių surišimas ir atlikimo eiliškumas

Jei mes sudėsime kodo dalis atsakingas už prezentacijos logiką (XHTML dokumentų generavimas), verslo taisyklių logiką ir duomenų prieigos logiką į vieną komponentą, tuomet mes turėsime vieno sluoksnio architektūrą (24 pav.).



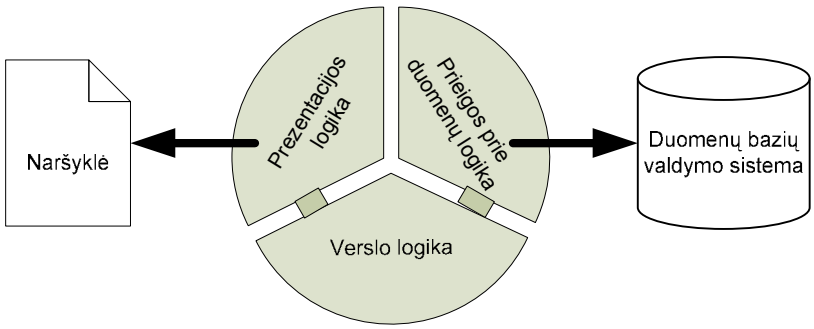
24 pav. Žiniatinklio aplikacijų kūrimo modelis naudojant tik vieno sluoksnio architektūrą

Atskyrus programinį kodą, kuris atsakingas už komunikaciją su fizine duomenų baze į atskirą komponentą, tuomet turėsime dviejų sluoksnių architektūrą (25 pav.).



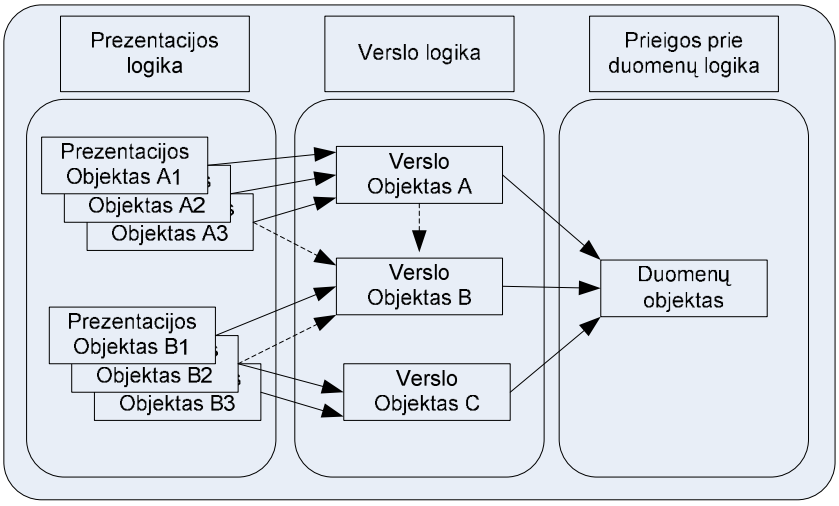
25 pav. Žiniatinklio aplikacijų kūrimo modelis naudojant dviejų sluoksnių architektūrą

Jei žengsime tolesnį žingsnį ir atskirsime prezentacijos logiką nuo verslo logikos – turėsime trijų sluoksnių architektūrą. Verta pabrėžti, kad šiame architektūros modelyje nėra tiesioginio komunikacinio ryšio tarp prezentacijos ir prieigos prie duomenų sluoksnio, visa informacija privalo pereiti per verslo sluoksnį (26 pav.).



26 pav. Žiniatinklio aplikacijų kūrimo modelis naudojant trijų sluoksnių architektūrą

Įgyvendinus žiniatinklio aplikaciją pagal mūsų siūlomą modelį, daugelis sistemos komponentų gali būti atkartotinai panaudami, išvengiama kodo dubliavimo. Komponentai esantys prezentacijos sluoksnyje gali naudoti tą patį verslo logikai priklausantį komponentą bei komponentai esantys verslo logikos sluoksnyje gali naudoti tą patį komponentą esantį prieigos prie duomenų sluoksnyje (27 pav.).



27 pav. Žiniatinklio aplikacijų kūrimo modelio komponentų sąveika

Mūsų siūlomo modelio apibendrinimas pateiktas 6 lentelėje.

6 lentelė. Siūlomo trijų sluoksnių modelio apibendrinimas

Modelio privalumai		**Modelio trūkumai**	
1.	Verslo logikos, atvaizdavimo ir duomenų atskyrimas vienas nuo kito, sudalinimas į tris savarankiškus sluoksnius;	1.	Sudėtingesnis proceso įsisavinimas;
2.	Vieno sluoksnio keitimas neįtakoja arba nežymiai paveikia kitus sluoksnius;	2.	Priklausomybė nuo papildomų bibliotekų.
3.	Atskirų sluoksnių realizavimas skirtingomis technologinėmis priemonėmis;	3.	Nėra detalių Smarty ir PEAR:DB integracijos aprašymų;
4.	Kiekvieno sluoksnio kūrimas gali būti atliktas skirtingų kūrėjų komandų;	4.	Prisirišama prie konkrečios technologijos, sunku atkartoti kitoje technologijoje.
5.	Atkartotinas kodo panaudojimas;	5.	
6.	Greitesnis kūrimo procesas;	6.	
7.	Platus panaudojamumas;	7.	
8.	Praplėtimo galimybės;	8.	
9.	Aukštesnis saugumo lygis.	9.	

3.4 Šablonų sistema Smarty - palaikanti kompiliavimą ir papildomus modulius

Smarty yra PHP kalbai skirta šablonų sistema sukurta Monte Ohrt ir Andrei Zmievski. Kyla klausimas ar Smarty yra eilinė šablonų sistema kaip daugelis kitų PHP kalbai skirtų šablonų sistemų? Autoriai (taip pat ir mes) su šiuo teiginiu nesutinka. Didžiausias skirtumas Smarty lyginant su kitomis populiariomis FastTemplate, patTemplate šablonų sistemos yra tas, kad Smarty geba tik vieną kartą interpretuoti šabloną - jį sukompiliuoja, todėl sekančiose užklausose šablonas nėra antrąkart interpretuojamas. Smarty puikiai atlaiko stresines apkrovimo situacijas, todėl dažnai naudojama didelio apkrovimo internetinėse aplikacijose.

Nors Smarty yra žinoma kaip šablonų sistema, ją tiksliau apbudintų terminas šablonų / prezentacijos karkasas. Žiniatinklio prezentacijos užduočių automatizavimui Smarty programuotojams ar šablono dizaineriams pateikia daugelį patogių įrankių. Karkasu ji vadinama, kadangi Smarty nėra vien tik šablono žymenis pakeičianti šablonų sistema. Nors ji

dažnai yra naudojama ir šiam paprastam tikslui, Smarty dėmesio centras yra greitas ir aiškus internetinės aplikacijos kūrimas, taip pat akcentuojant keičiamumą, saugumą ir nuolatinį augimą.

3.4.1 Svarbiausios Smarty savybės

3.4.1.1 Šablonų kompiliavimas

Smarty interpretuoja šabloną ir po to sukuria PHP skriptą (vietoje binarinės versijos, kaip įprasta programavime). Kuomet internetinis puslapis yra peržiūrimas, Smarty naudoja iš šablono sukurtą PHP skriptą, vietoj to, kad šabloną analizuotų dar kartą, todėl yra išvengiamas pakartotinas šablono interpretavimas. Smarty kompiliavimo procesas yra gana įmantrus: pakartotinas kompiliavimas yra atliekamas tik tada, jei yra pakeičiamas šablonas, todėl nereikia rūpintis šablono kompiliavimo procesu – jis yra automatizuotas.

Visas kompiliavimo procesas yra paslėptas. Šablono dizaineris atlikdamas šablono pakeitimus neprivalo, netgi žinoti ar Smarty yra kompiliavimą palaikanti šablonų sistema. Pakeistus šabloną, automatiškai pagal jį bus sukurtas PHP skriptas.

3.4.1.2 Šablonų kešavimas

Smarty nustatymam laiko tarpui gali apdorotą šablono rezultatą padėti į tarpinę saugyklą (*cache*), todėl nustatytą periodą nebus atliekamas duomenų nuskaitymas iš duomenų šaltinio. Dažnai duomenų šaltinis yra išorėje bei prieiga prie jo yra lėta, tai - viena lėčiausių aplikacijos vietų. Panaudojant tarpinę saugyklą Smarty laikinai išsaugo šablono rezultatą su nuskaitytais duomenimis iš duomenų šaltinio, kuomet atliekama prieiga prie internetinio puslapio, prisijungimas prie duomenų šaltinio neatliekamas, rezultatai pasiimami iš tarpinės saugyklos. Jei turime lėtai į užklausas atsakančią duomenų bazių valdymo sistemą arba ji yra apkrauta didelių kiekiu užklausų, tarpinės saugyklos panaudojimas žymiai pagerins internetinės aplikacijos greitį bei atsakomumą.

Egzistuoja atvejų kada nėra tikslinga naudoti Smarty tarpinę saugyklą (*cache*). Pavyzdžiui, internetinės aplikacijos, kuriose duomenų bazės duomenis keičiasi labai dažnai ir reikalingas tikslus duomenų bazės informacijos atspindėjimas internetiniame puslapyje. Bet ir šioje vietoje Smarty yra gana lanksti, ji įgalina nurodyti ką norima patalpinti į tarpinę saugyklą, o ką ne.

Panaudojant tarpinę saugyklą atskiroms šablono dalims galima pasiekti „aukso vidurį“ tarp naujausio dinaminio turinio ir greito puslapių užkrovimo.

3.4.1.3 Šablonų kintamųjų modifikatoriai

Smarty pateikia kintamųjų modifikatorius, kurie yra naudojami šablono kintamųjų turiniui modifikuoti. Naudojant modifikatorių galima šablono kintamojo turinį perversti į didžiąsias raides (pvz. $title|upper), sutrumpinti simbolių eilutę (pvz. $content|truncate:30 šis modifikatorius parodys pirmus 30 $content kintamojo simbolių), panaudoti reguliarias išraiškas simbolių eilučių paieškai ar pakeitimui (pvz. $article|regex_replace:"/bad word/":"***" modifikatorius pakeis visas kintamajame $article esančias išraiškas "bad word" į "***"), suformuoti datas, valdyti HTML kodo atvaizdavimą.

Kintamųjų modifikatoriai suteikia galimybę šablonų dizaineriams atlikti šablono kintamųjų turinio modifikavimą be didesnių programavimo subtilybių. Šis programavimo atspindys šablono dizaineriams suteikia didesnę formatavimo kontrolę, nors jie ir turi sužinoti egzistuojančius modifikatorius bei jų funkcionalumą.

3.4.1.4 Šablono funkcijos

Šablono dizaineriai gali pasinaudoti standartinėmis šablono funkcijomis bei susikurti jas patys. Šios funkcijos yra tarsi Smarty šablonų API. Funkcijos įgalina atlikti išvedimus pagal tam tikrą sąlygą (if išraiškos), įvykdyti dinaminių blokų iteracijas (naudojant foreach, section), aktyvuoti konfigūracijos failus, realizuoti ciklus, generuoti HTML kodo segmentus (lentelės, meniu) ir daug daugiau.

3.4.1.5 Šablono filtrai

Smarty įgalina programuotojus panaudoti filtrus prieš ar po šablono kompiliavimo. Prieš-filtrai yra apdorojami prieš šablono kompiliavimą. Po-filtrai – po šablono kompiliavimo. Išvedimo filtrai – gavus išvedimo užklausą.

Kyla klausimas kam reikalingas filtrų panaudojimas? Naudojant prieš-filtrus galima pašalinti nepageidaujamus komentarus (pvz. sukurtus programavimo IDE) bei garantuoti, kad kompiliatorius gaus norimą šablono turinį. Panaudojant po-filtrus po šablono kompiliavimo galima šabloną papildyti papildoma informacija, tokia kaip šablono sukūrimo data, autorius, versija ir pan. Išvedimo filtrai suteikia galimybę modifikuoti išvedamą šablono turinį, todėl galima atlikti elektroninio pašto adresų užmaskavimą apsaugant nuo nepageidaujamų laiškų gavimo (*spam*) (panaudojant preg_replace() reguliarias išraiškas).

3.4.1.6 Šablono konfigūraciniai failai

Šablono konfigūracijos failuose yra saugomi globalūs šablono kintamieji, kurie gali daryti poveikį daugiau nei vienam šablonui. Pavyzdžiui, globalūs kintamieji gali būti panaudoti šablono spalvinei schemai saugoti. Pakeitus globalius spalvų kintamuosius,

pakeitimai bus aplikti visuose šablonuose, kuriuose yra naudojamas konfigūracinis failas, aprašantis spalvų schemas.

3.4.1.7 Šablono papildiniai (*plugins*)

Smarty papildinių architektūra buvo pristatyta 2.0 versijoje, tai įgalino Smarty prisitaikyti pagal individualius poreikius. Naudojant papildinius galima susikurti savo šablono funkcijas, kintamųjų modifikatorius ir filtrus. Pasitelkiant pagalbinius papildinius šablonų saugojimas gali būti pakeistas iš failų į duomenų bazę.

3.4.1.8 Papildomi moduliai

Didelį indėlį į papildomų modulių kūrimą įnešė Smarty vartotojų bendruomenės, pasiūlydamos tokius papildomus modulius kaip puslapiavimas, formų validacija, kalendoriaus datos pasirinkimas ir daugelį kt. Pasinaudojus papildomais moduliais galima žymiai pagreitinti žiniatinklio sistemų kūrimą, nes nebereikia iš naujo išrasti to kas yra sukurta kitų, taip pat pasirinktam moduliui nereikalingas papildomas testavimas.

3.4.2 Smarty šablonų sistemos apibendrinimas

Gerai suderinus PHP programavimo technikas ir Smarty šablonų sistemą galima pasiekti didelį žiniatinklio puslapių generavimo našumą. Taip pat Smarty pateikia išplėstą funkcionalumą, įskaitant šablono funkcijas ir kintamųjų modifikatorius, kurie gali būti praplečiami panaudojant papildinius.

Smarty panaudojimas siejamas ir su didesniu saugumo lygmeniu, kadangi šablone nėra naudojamas PHP kodas. Šablono dizaineris neturi galimybių prieiti prie visiškos PHP kontrolės, jis tik atsakingas už ribotą funkcionalumą, kurį apibrėžia programuotojas.

Kadangi programuotojas atsakingas už verslo logiką, o dizaineris už prezentaciją, tokią internetinę aplikaciją yra lengvą palaikyti, atlikti pakeitimus. Keičiant vartotojo sąsaja, programuotojui nereikės atlikti sudėtingų keitimų, jam gali reikėti atlikti tik nežymius pakeitimus verslo srityje.

Smarty yra ypatingai naši šablonų sistema, nors ji savo funkcionalumu, plačiu savybių ratu sunkiai pralenkiama. Daugelis Smarty funkcijų yra atliekama per papildinių palaikymą, kurie esant reikalui gali būti įjungiami arba išjungiami. Šablonų kompiliavimas taip pat viena iš našumą pagerinančių priemonių, kuri suteikia galimybę išvengti pakartotinų šablono interpretavimų.

Geros Smarty greičio ir funkcionalumo savybės nesumažinama ir vartojamumo: mokymosi sudėtingumo kreivė nėra aukštesnė už kitų šablonų sistemų. Mokymo procesą

palengvina Smarty kūrėjų pateikiama dokumentacija, kuri yra visapusiška ir lengvai suprantama.

Reikia pažymėti, kad Andrei Zmievski yra vienas iš Smarty autorių, kuris taip pat priklauso ir PHP kūrėjų komandai, todėl Smarty kūrimo procesas atrimai siejasi su PHP. Naujausi PHP pakeitimai greitai adaptuojami Smarty šablonų sistemai.

3.5 PEAR::DB - duomenų abstrakcijų klasė

Vienas iš didžiausių PHP programavimo kalbos privalumų yra lengvas prieigos prie duomenų bazių skriptų rašymas, kurių pagalba sukuriami dinaminiai žiniatinklio puslapiai inkorporuojantys duomenų bazės turinį. Tai yra svarbu kada reikia lankytojams pateikti naujausią duomenų bazės informaciją, vietoj to, kad rankiniu būdu redaguoti statinius HTML puslapius ar PHP duomenų masyvus. Nors PHP yra lengva naudoti, tačiau ji neturi bendros prieigos prie duomenų bazių sąsajos. PHP kalboje naudojami atskiri funkcijų poaibiai kiekvienai duomenų bazių valdymo sistemai. Vienas poaibis yra skirtas MySQL duomenų bazių valdymo sistemai, kitas - InterBase, ir t.t. Platus duomenų bazių valdymo sistemų palaikymas, padaro PHP programavimo kalbą populiarią, nesvarbu kokią DBVS bus naudojama, PHP tikriausiai ją palaikys. Esant skirtingiems kiekvienos DBVS poaibiams, PHP skriptai leksiniame (programinio kodo) lygyje nėra portatyvūs.

Pavyzdžiui, funkcijos pateikiančios SQL užklausas vadinamos mysql_query(), ibase_query(), pg_exec() priklausomai nuo naudojamos DBVS: MySQL, InterBase, PostgreSQL. Prireikus panaudoti kitą DBVS ar įdiegti PHP kalba parašytą internetinę aplikaciją, kuri naudoja kitą DBVS sistemą negu esama, priverstinai reikės visus skriptus pritaikyti prie kitos DBVS.

Šią problemą galima išspręsti pasinaudojus PHP išplėtimų ir priedų saugykla (*PEAR*). Iš PEAR saugyklos pasinaudosime PEAR::DB dvejų lygių architektūros duomenų abstrakcijos klase:

- Aukščiausiame lygyje pateikiama abstrakti sąsaja, kuri yra vienoda visoms PEAR::DB palaikomoms DBVS. Ši sąsaja taip pat paslepia specifinę DBVS detalizaciją. Programuotojams nereikia rūpintis pernešamumo problemomis, kadangi visoms DBVS pateikiamas tik vienas abstraktus prieigos prie DBVS funkcijų poaibis.
- Žemas lygis yra sudarytas iš individualių tvarkyklių. Kiekviena tvarkyklė palaiko konkrečia DBVS bei interpretuoja abstrakčią sąsają, pagal kurią parenka DBVS specifinę sąsają.

PEAR::DB pateikia abstrakčią, objektiškai orientuotą programavimo sąsają (*API*) trylikai pagrindinių DBVS: dBase, FrontBase, InterBase/Firebird, Informix, mSQL, MS SQL Server, MySQL, MySQLi, Oracle, ODBC (testuota DB2 ir MS Access), PostgreSQL, SQLite bei Sybase.

3.5.1 Portatyvumo svarba

Yra daug sistemos kūrėjų, kurie portatyvumui neskiria didelio dėmesio. Atvejai kada yra pakeičiamas egzistuojančiuos internetinės aplikacijos DBVS į kitą yra reti, tačiau tokie atvejai kaip nauji licencijuos terminai, padidėjusi kaina, pasikeitę poreikiai gali priversti pereiti prie kitos DBVS. Viena svarbiausių priežasčių lėmusių didelį duomenų bazių abstrakcijų klasių populiarumą yra lengvas sukurtos sistemos pritaikymas plačiam sistemų ratui.

3.5.2 Portatyvumo rėžimai

Kiekviena DBVS turi savų funkcionalumo bruožų, todėl visos DBVS yra įvairiais aspektais subtilios, skirtingos. Pavyzdžiui, vienos duomenų bazių valdymo sistemos išvesdamos lentelės laukų pavadinimus juos perverčia į didžiąsias raides, kitos - į mažąsias arba palieka nekeistas. Dėl šių keistybių yra sunku perrašyti skriptus kito tipo duomenų bazių serveriui. Daugelį esančių nesuderinamų tarp skirtingų DBVS, PEAR::DB sėkmingai įsprendžia, dažniausiai perėjimas prie kitos DBVS atliekamas be papildomų pakeitimų.

Portatyvumo savybės yra nustatomos panaudojant *portability* konfigūracinę direktyvą. Iš viso yra aštuoni portatyvumo rėžimai, jie pateikiami 7 lentelėje. Kiekvienas rėžimas yra reprezentuojamas bitine konstantą, todėl jos gali būti apjungiamos kartu, panaudojant bitines operacijas (| arba ^)>

7 lentelė. Portatyvumo rėžimai

Portatyvumo rėžimas	Aprašymas
DB_PORTABILITY_ALL	Įjungiamos visos portatyvumo savybės
DB_PORTABILITY_DELETE_COUNT	Vertimas skaičiuoti ištrintų eilučių kiekis
DB_PORTABILITY_ERRORS	Tam tikrų klaidų pranešimų, suderinimas su visom DBVS
DB_PORTABILITY_LOWERCASE	Lentelių pavadinimų bei laukų pavadinimų pervertimas į mažąsias raides
DB_PORTABILITY_NONE	Išjungiamos visos portatyvumo savybės
DB_PORTABILITY_NULL_TO_EMPTY	NULL reikšmių konvertavimas į tuščią simbolių eilutę
DB_PORTABILITY_NUMROWS	numRows() palaikymas su Oracle duomenų bazių valdymo sistemomis
DB_PORTABILITY_RTRIM	Tarpų simbolių pašalinimas iš eilutės galo, paveikia funkcijas get*() and fetch*()

3.5.3 PEAR::DB duomenų abstrakcijų klasės apibendrinimas

Uždedant abstrakcijos sluoksnį ant standartinių bei kiekvienai DBVS skirtingų funkcijų aibės yra suvienijama prieigos prie DBVS sąsaja. Programuotojams yra žymiai paprasčiau parašyti lengvai pernešamą kodą, tačiau auga struktūrinis aplikacijos sudėtingumas. Pagrindinis PEAR::DB minusas – žemajame lygyje yra naudojamos standartinės PHP prieigos prie DBVS funkcijos, PEAR::DB žemasis lygmuo nėra perrašytas į gimtąjį C kodą. Dėl šio trūkumo yra prarandamas, nors ir nedidelis, produktyvumas.

4 STUDENTŲ PRAKTIKOS DARBŲ PORTALO PROJEKTAS

4.1 Sistemos paskirtis

Efektyvi darbininkų samdymo strategija yra svarbi kiekvienai organizacijai. Šiuo metu padidėjęs Interneto technologijų naudojimas verčia kurti bei naudoti internetinius darbininkų samdymo sprendimus. Projektas skirtas internetinio studentų įdarbinimo portalo sukūrimui.

Įgyvendintas projektas palengvintų šiuo metu aktualią įdarbinimo problemą. Studentai portalu naudotis būtų skatinami dėstytojų, praktikos darbo pasirinkimui bei pačių studentų patogumui sukuriama galimybė susirasti darbą, o tuo tarpu įmonės turėtų didelį privalumą susirasti pigią kvalifikuotą darbo jėgą. Galutinio produkto vartotojai bus studentai, dėstytojai, įmonės atstovai ir sistemos administratoriai.

4.2 Projekto tikslas

Projekto tikslas - sukurti informatyvų, lengvai naudojamą, valdomą, saugų internetinį portalą. Internetinio tinklapio pagalba įmonės turėtų galimybę siūlyti darbus, susirasti tinkamą studentą, o studentai galėtų rasti sau tinkamą praktikos darbą. Kaip parodė dėstytojų apklausa šiuo metu praktikos darbų pasirinkimas ir kontrolė nėra trivialus dalykas. Dėstytojai ir studentai susiduria su problemomis: nedidelis firmų aktyvumas įdarbinimo procese, maža norimo darbo pasirinkimo galimybė, maža studentų kontrolė.

Pagrindinės funkcijos, kurias turi užtikrinti sukurtas portalas:

- Palengvinti ir automatizuoti praktikos darbų pasirinkimą. Būsimieji vartotojai – studentai;
- Palengvinti ir automatizuoti praktikos darbų valdymą. Būsimieji vartotojai – dėstytojai;
- Suteikti galimybes įmonės atstovams pasiūlyti praktikos darbus. Padėti jiems susirasti kvalifikuotą darbo jėgą;
- Sistema turi suteikti galimybes sistemos vartotojams bendrauti tarpusavyje;
- Sistema turi užtikrinti įvairių lygių priėjimo prie duomenų teises.

Portalo savybės:

- Informatyvumas;
- Lengvas naudojimasis;

Portalo paskirtis:

- Įmonės atstovui pasiūlyti darbo temą bei susirasti tinkamą studentą užduoties atlikimui;
- Studentui susirasti praktikos darbą;

- Dėstytojui palengvinti studentų ir firmų kontrolę.

Vartotojai:

- Studentai;
- Dėstytojai;
- Įmonės atstovai;
- Administratoriai.

4.3 Įpareigojantys apribojimai

4.3.1 Apribojimai sprendimui

- Portalas kuriamas panaudojant šablonų sistemą Smarty ir duomenų abstrakcijos klasę PEAR::DB;
- Klientinė sistemos pusė turi būti nepriklausoma nuo platformos. Klientas turi veikti šiuose operacinių sistemų šeimose: Linux, Unix, Windows, Mac;
- Kliento pusė bus palaikoma interneto naršyklės pagalba, kuri suderinama su XTHML 1.0 ar vėlesniu standartu;
- Serverio pusė privalo veikti visose Unix ir Linux šeimų operacinėse sistemose;
- Sistema turi naudoti ISO-8859-13 simbolių kuoduotę;
- Sistemoje turi būti realizuota vartotojų teisių sistemą, kuri leis suteikti skirtingiems vartotojams, skirtingas privilegijas;
- Produkto klientinė pusė privalo veikti delniniame kompiuteryje, kuriame palaikoma XHTML naršyklė.

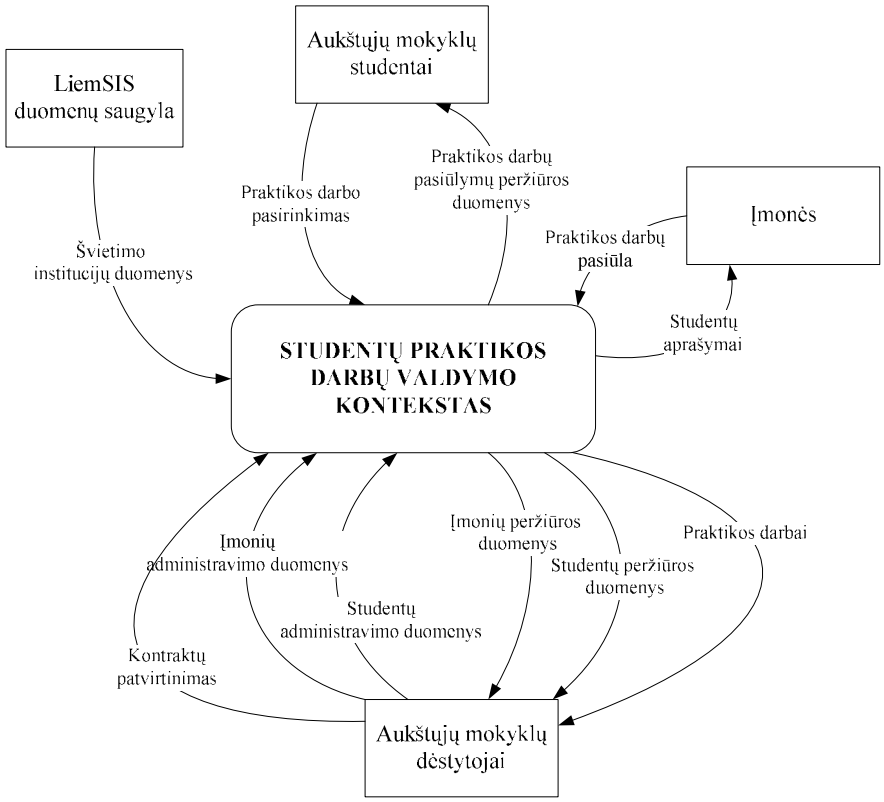
4.3.2 Bendradarbiaujančios sistemos

Produktas kuriamas klientas – serveris modeliu. Todėl yra daug programinių priemonių kurios nėra kuriamo produkto dalis. Tarpusavyje bendradarbiaujančios sistemos pateikiamos 8 lentelė:

8 Lentelė. Tarpusavyje bendradarbiaujančios sistemos

Programinės įrangos tipas	Rekomenduojama programinė įranga	Bendradarbiauja su	Papildoma informacija
Naršyklė	FireFox	Web serveriu	Vartotojai sistemą valdo naršyklės pagalba.
Web serveris	Apache	Naršykle ir Web serviso moduliu	Bendravimas su naršykle
Web serviso modulis	PHP	Naršykle, Lokalia DB ir išorine sistema LiemSIS	Programinių modulių interpretavimas. Varotojo duomenų apdorojimo variklis
Lokali DB	PostgreSQL	Web serviso moduliu	Informacijos saugykla
LiemSIS DB	-	Web serviso moduliu	Egzistuojanti Lietuvos aukštųjų mokyklų duomenų saugykla.

4.4 Veiklos sudėtis



28 pav. Studentų praktikos darbų valdymo konteksto diagrama

4.5 Sistemos ribos

Aktorių sąrašas: Dėstytojas, studentas, įmonės atstovas, sistemos administratorius, duomenų saugyklos.

Panaudojimo atvejai: Praktikos darbų registravimas, Praktikos darbų patvirtinimas, Praktikos darbų modifikavimas, Praktikos darbų pasirinkimas, Praktikos darbų peržiūra, Vartotojų valdymas, Vartotojų peržiūra, Bendravimas tarp vartotojų, Sistemos administravimas.

Aktorių funkcijos:

1. Dėstytojas: asmuo kontroliuojantis sistemos vartotojus, susitarimus tarp studentų ir įmonės atstovų. Gali peržiūrėti, pasiūlyti, tvarkyti praktikos darbus.

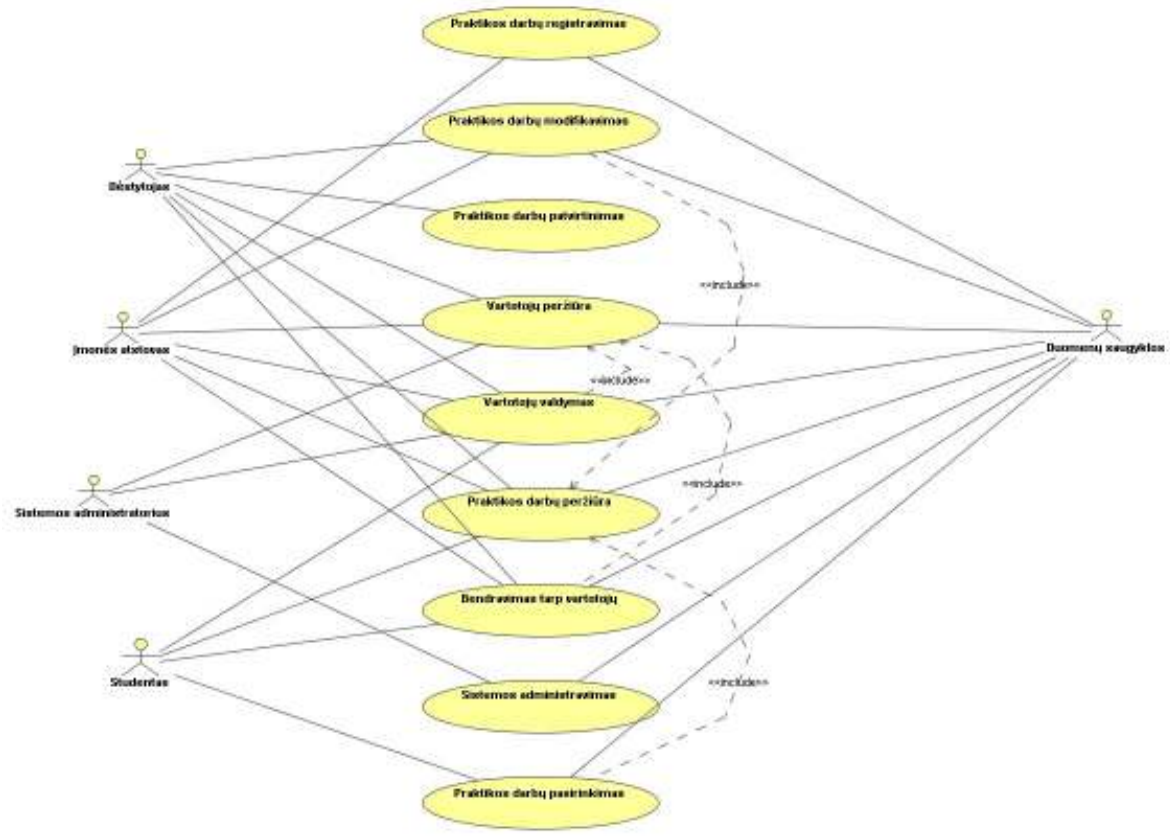
2. Įmonės atstovas: Priregistruoti savo firmą į sistemą, pasiūlyti temas studentams, diskutuoti su studentais.

3. Studentas: Prisiregistruoti į sistemą, peržiūrėti siūlomus firmos praktikos darbus, diskutuoti su firmomis dėl dominančios temos, pasirinkti praktikos darbą.

4. Sistemos administratorius: Valdyti sistemą, užtikrinti saugumą, sistemos našumą, patikimumą, perspėti sistemos vartotojus.

5. Duomenų saugyklos sistema: Tai sistema kur saugomi, taip pat iš kur pasiimami

duomenys. Duomenų saugykla susideda iš LiemSIS duomenų bazių ir savos duomenų saugyklos.



29 pav. Panaudojimo atvejų diagrama

Detalus panaudojimo atvejų sąrašas pateiktas 2 priede.

4.6 Rizikos

4.6.1 Sistemos kūrimo rizikos

9 Lentelė. Sistemos kūrimo rizikos

Nr.	Rizikos faktorius	Tikimybinis įvertinimas
1.	Perdėtai suspaustas kūrimo grafikas	0,5
2.	Vadovavimo praktikos stoka	0,8
3.	Projekto nutraukimas	0,1
4.	Papildomi reikalavimai	0,7
5.	Lėtas užsakovo reikalavimų pateikimas	0,8
6.	Architektūros pasikeitimas	0,6
7.	Projekto vėlavimas	0,4
8.	Žema kokybė	0,5
9.	Žemas produktyvumas	0,5

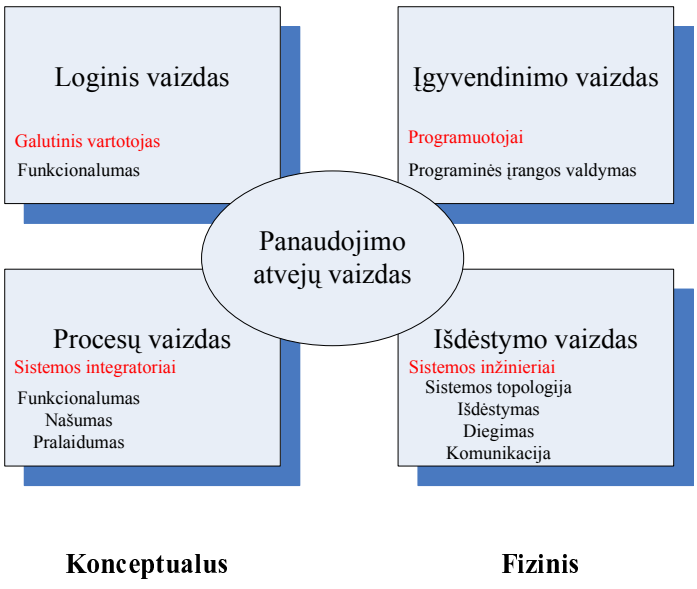
4.6.2 Atsitiktinumų (rizikų) planas

10 Lentelė. Rizikos faktoriai ir problemų sprendimai

Nr.	Rizikos faktorius	Problemos sprendimas
1.	Perdėtai suspaustas kūrimo grafikas	Planuoti atsižvelgiant į visus faktorius. (Detalus visų darbų planavimas)
2.	Vadovavimo praktikos stoka	Patyrusių specialistų konsultacijos.
3.	Projekto nutraukimas	Potencialių klientų paieška.
4.	Papildomi reikalavimai	Gerai išsiaiškinti vartotojo reikalavimus.
5.	Lėtas užsakovo reikalavimų pateikimas	Užsakovo motyvavimas pateikti reikalavimus.
6.	Architektūros pasikeitimas	CASE įrankių panaudojimas, papildomo laiko numatymas.
7.	Projekto vėlavimas	Plano koregavimas, pasitarimas su užsakovu.
8.	Žema kokybė	Naudoti gerai žinomas technologijas. Naudotis automatizuotais testavimo paketais.
9.	Žemas produktyvumas	Produktyvių įrankių pasirinkimas.

4.7 Praktikos darbų portalo architektūra

Architektūros specifikacija yra parengta *Rational Unified Process* (*RUP*) pagrindu, pasinaudojant *Unified Modeling Language* (*UML*). Sistemos architektūrai atvaizduoti yra naudojami pagrindiniai penki vaizdai: panaudojimo atvejų, loginis, procesų, įgyvendinimo ir diegimo (30 pav.).

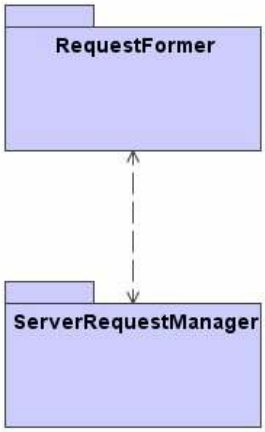


30 pav. Sistemos architektūros reprezentacija

4.7.1 Sistemos paketų diagramos

Studentų praktikų ir baigiamųjų darbų portalo architektūros aukščiausio lygio patekai pateikti 31 pavyzdyje. Aukščiausio lygio paketų paskirtis:

- „RequestFormer“ – Sąsaja kurios pagalba yra valdoma sistema, formuojamos užklausos serveriui;
- „ServerRequestManager“ – Užklausų bei operacijų apdorojimo valdiklis;



31 pav. Studentų praktikų ir baigiamųjų darbų portalo architektūros aukščiausio lygio paketai

„RequestFormer“ paketas yra sudėtinis (32 pav.), jį detalizuoja šie paketai:

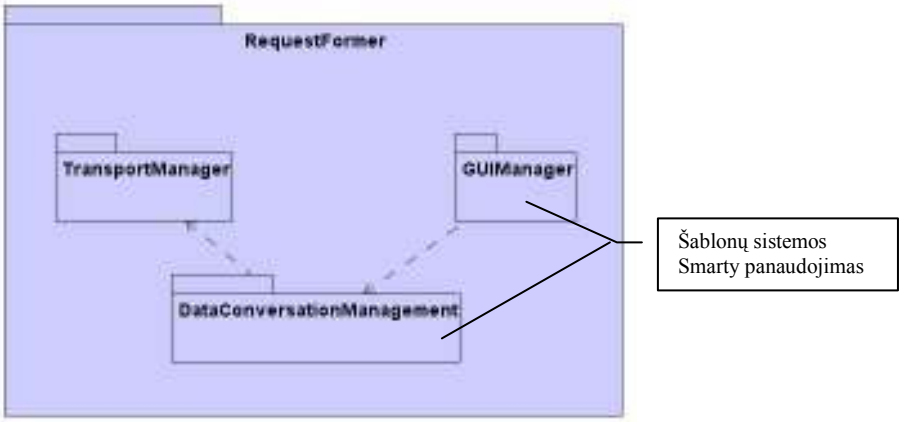
- „GUIManager“ – paketas atsakingas už vartotojo grafinės sąsajos funkcionalumą. Šis paketas sąveikauja su „DataConversationManagement“ paketu. Panaudojant Smarty šablonų sistemą, duomenys priskirti šablono kintamiesiems yra atvaizduojami.
- „DataConversationManagement“ – duomenų pritaikymas grafinei sąsajai bei rezultatų perdavimas serveriui. Paketas sąveikauja su „TransportManager“ paketu. Panaudojant Smarty šablonų sistemą, duomenys yra priskiriami šablono kintamiesiems.
- „TransportManager“ – paketas atsakingas už duomenų mainus tarp serverio ir kliento. Šio paketo dėka yra užmezgamas ryšys su serveriu.

„ServerRequestmanager“ paketas yra sudėtinis (33 pav.), jį detalizuoja šie paketai:

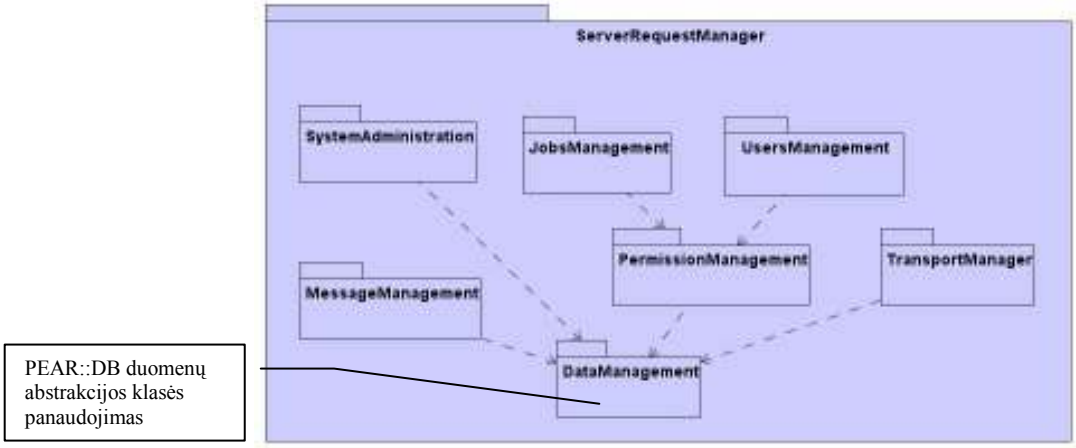
- „SystemAdministration“ – paketas atsakingas už sistemos apvalymą, saugumą, stabilumo užtikrinimą. Paketas sąveikauja su „DataManagement“ paketu.
- „JobManagement“ – paketas atsakingas už praktikos darbų įrašymą, patvirtinimą, pašalinimą bei pasirinkimą. Paketas sąveikauja su „PermissionManagement“ ir „DataManagement“ paketu.
- „UserManagement“ – paketas atsakingas už vartotojų kūrimą, valdymą, peržiūrą, šalinimą. Paketas sąveikauja su PermissionManagement“ ir „DataManagement“ paketu.

- „PermissionManagement“ – leidimų valdymas. Paketas sąveikauja su „DataManagement“ paketu.
- „MessageManagement“ – paketas atsakingas už žinučių valdymą. Šio paketo dėka vartojai gali rašyti vieni kitiems žinutes, jas skaityti bei pašalinti. Paketas sąveikauja su „DataManagement“ paketu.
- „TransportManager“ – paketas atsakingas už duomenų mainus tarp serverio ir kliento. Šio paketo dėka yra užmezgamas ryšys su serveriu. Paketas sąveikauja su „DataManagement“ paketu.
- „DataManagement“ – paketas atsakingas už informacijos išsaugojimą, informacijos apdorojimą. Pakete esančios klasės su duomenų bazių valdymo sistemomis bendrauja naudojant PEAR:DB duomenų abstrakcijos klasę, kuri suteikia unifikuotą, portatyvę sąsają.

Aukščiausio lygio paketai „RequestFormer“ ir „ServerRequestmanager“ vienas su kitu susirašą per žemesnio lygio paketą „TransportManager“.



32 pav. „RequestFormer“ paketo sudėtis

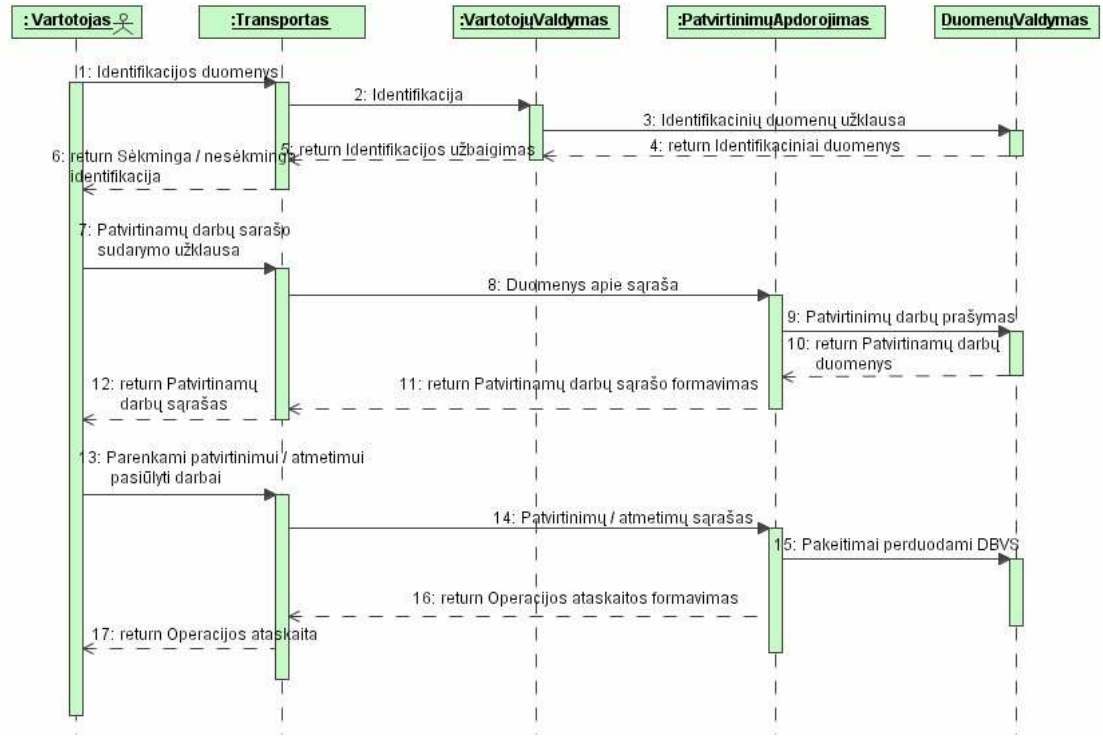


33 pav. „ServerRequestmanager“ paketo sudėtis

4.7.2. Procesų vaizdas

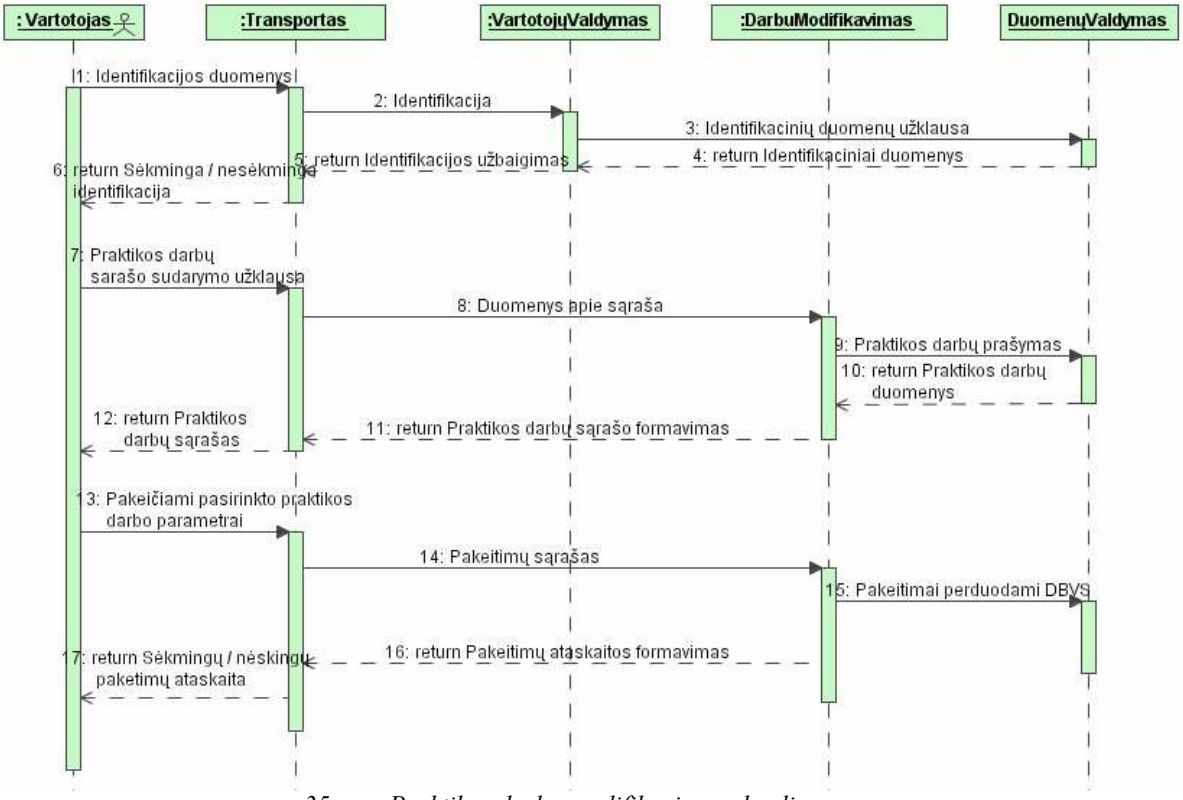
Pateikiamos tik pagrindinės sąveikos sekų (*sequence*) ir būsenų (*state*) diagramos.

Praktikos darbų patvirtinimo sekų diagrama



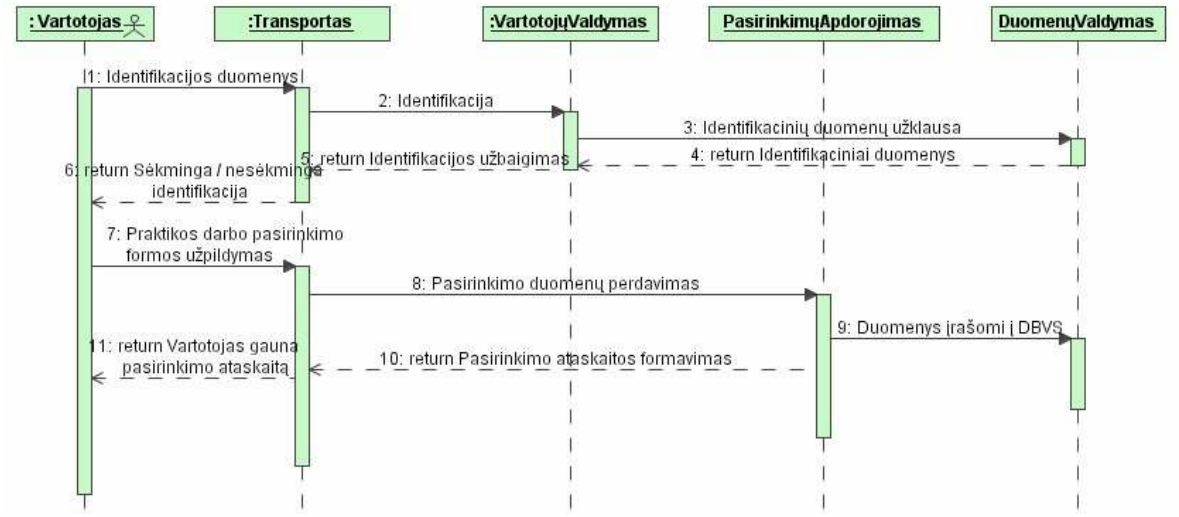
34 pav. Praktikos darbų patvirtinimo sekų diagrama

Praktikos darbų modifikavimo sekų diagrama



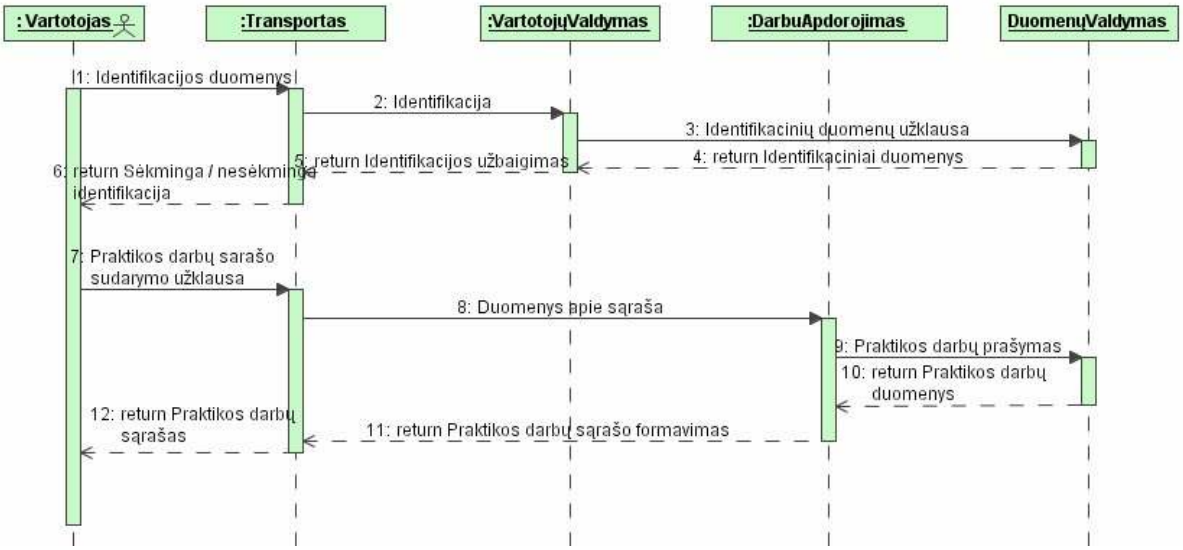
35 pav. Praktikos darbų modifikavimo sekų diagrama

Praktikos darbų pasirinkimo sekų diagrama



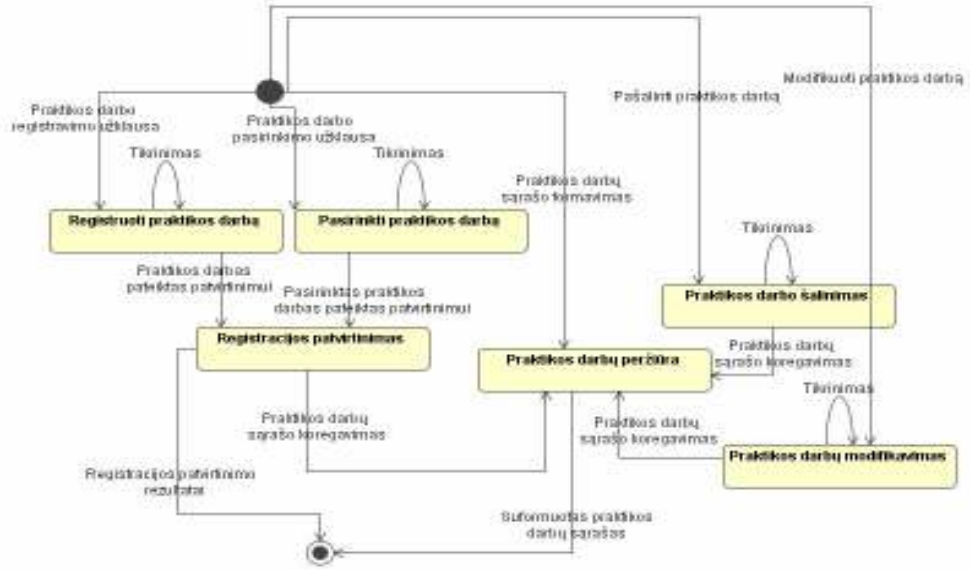
36 pav. Praktikos darbų pasirinkimo sekų diagrama

Praktikos darbų peržiūros sekų diagrama



37 pav. Praktikos darbų peržiūros sekų diagrama

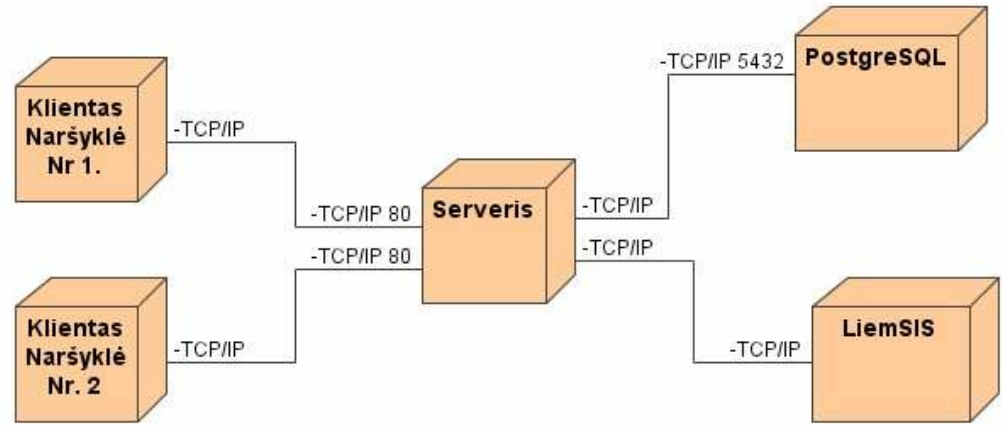
Praktikos darbų valdymo būsenų diagrama



38 pav. Praktikos darbų valdymo būsenų diagrama

4.7.4 Išdėstymo vaizdas

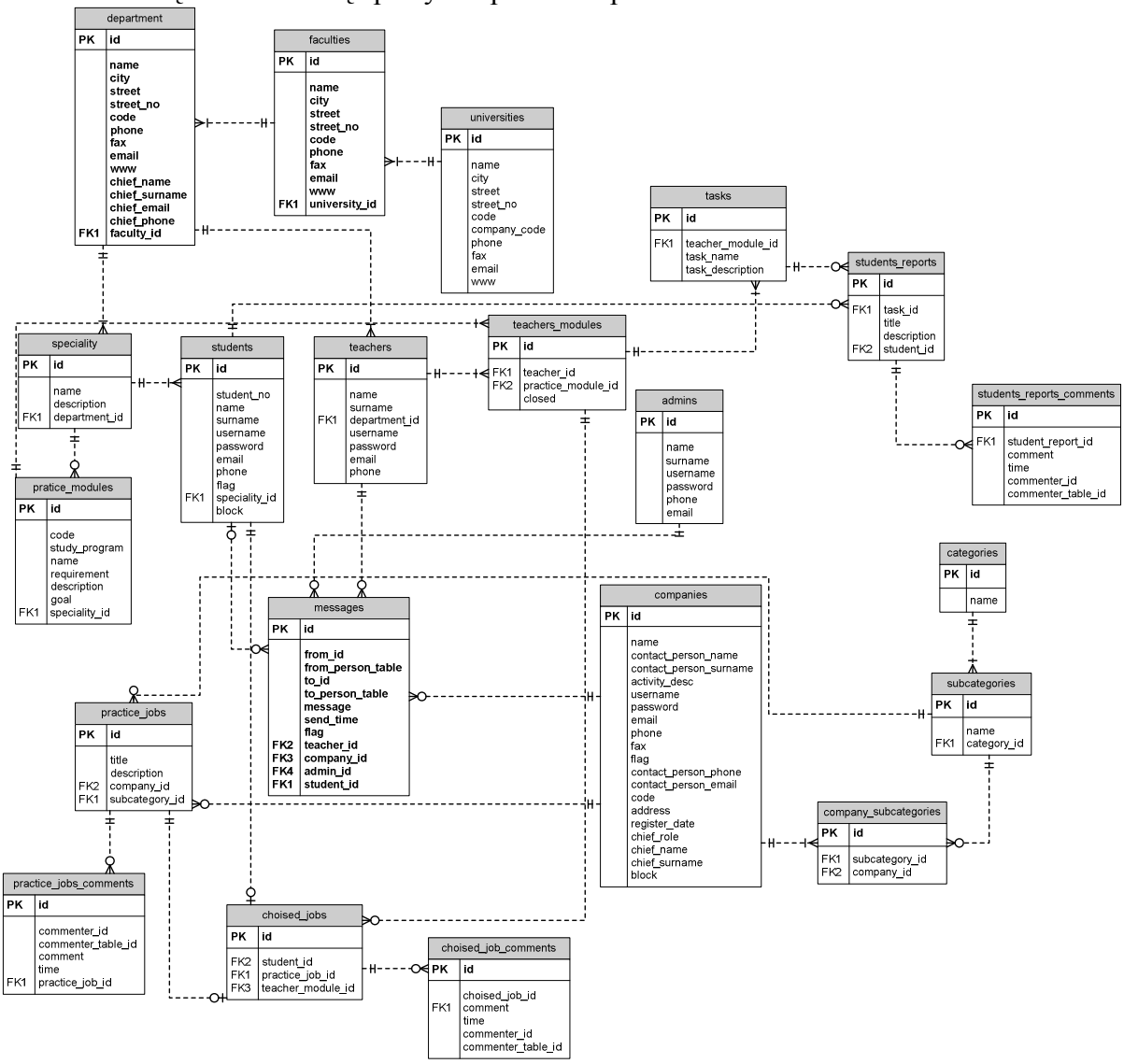
Sistemos išdėstymo vaizdas pateiktas (39 pav.).



39 pav. Išdėstymo vaizdas

4.7.5 Duomenų bazės vaizdas

Duomenų bazės lentelių aprašymai pateikti 3 priede.



40 pav. Detali duomenų bazės schema

4.8 Projekto kokybės vertinimas

4.8.1. Funkcionalumo tyrimas

Tyrimo eigoje sukurtu portalu naudojosi įvairaus lygio vartotojai. Buvo siekiama patikrinti sistemos funkcionalumo atitikimą vartotojo reikalavimams. Vartotojams tyrimo metu buvo pateiktas sistemos funkcijų sąrašas, kuriame jie pažymėjo programų sistemos atitikimą specifikacijai (11 lentelė). Išpildyti reikalavimai pažymėti pliusu, o atskiriems panaudojimo atvejams, nenumatyti veiksmai, pažymėti minusu.

A – Administratorius

S – Studentas

Į – Įmonė

D - Dėstytojas

11 lentelė. Sistemos funkcionalumas

Reikalavimų nr.	**Funkcionalumas**	**Aktoriai**			
		A	S	Į	D
1,2,3	Praktikos darbų registravimas	+	-	-	-
5	Praktikos darbų patvirtinimas	-	-	-	+
7,8,9	Praktikos darbų modifikavimas	-	-	+	+
11,12	Praktikos darbų pasirinkimas	-	+	-	-
13	Praktikos darbų peržiūra	-	+	+	+
15,16,18,19	Vartotojų valdymas	+	+	+	+
14,17	Vartotojų peržiūra	+	+	+	+
20,21,22,23	Bendravimas tarp vartotojų	-	+	+	+
29,30,31,32	Sistemos administravimas	+	-	-	-

Gauti tyrimo rezultatai patvirtino programų sistemos funkcionalumo atitikimą sistemos specifikacijai.

4.8.2 Korektiškumas – tai kokybės faktorius atspindintis programos atitikimą specifikacijai ir vartotojo poreikių tenkinimą. Buvo sukurti keli prototipai papildomų reikalavimų išgavimui bei vartotojo sąsajos pobūdžio nustatymui Ji atitinka visus apibrėžtus reikalavimus ir visiškai tenkina vartotojo poreikius apibrėžtose srityse.

4.8.3 Patikimumas – Sistema yra apsaugota nuo nepatikimų vartotojų veiksmų:

- Duomenų bazės lentelės yra surištos tam tikromis taisyklėmis. Pavyzdžiui administratoriui neleidžiama pašalinti universiteto, kol nepašalinti visi fakultetai esantys tame universitete;
- Neleidžiama įvesti nekorektiškų duomenų. Jei informacinis laukas susideda tik iš skaitmenų, tai vartotojas galės įvesti tik skaitmenis, jei iš raidžių – tik raides ir pan.

4.8.4 Saugumas nusako autorizuotą priėjimą prie programinės įrangos ar jos duomenų. Kiekvienas sistemos vartotojas turi tik autorizuotą priėjimą prie sistemos, jos duomenų bei veiksmų su sistema. Kiekvienas sistemos vartotojas turi prisijungimo vardą ir slaptažodį, kuriais prisijungęs prie sistemos, gali dirbti tik su jam reikalingais duomenimis ir atlikti tik jam pavestas funkcijas. Todėl saugumas vertinamas kaip aukštas.

4.8.5 Efektyvumas apibrėžia santykį tarp kompiuterinių resursų, reikalingų programinei įrangai funkcionuoti ir programinės įrangos įmonėms, universitetams teikiamos naudos. Serveryje įdiegtas portalas užima apie 5 MB vietos kietame diske. Papildomai užimama vieta priklauso nuo duomenų bazės dydžio. Palyginti su turimais resursais, programinės įrangos naudojami resursai nėra dideli. Ši informacinė sistema žymiai sumažina įmonėms darbuotojų paieškos, dėstytojams praktikos darbų valdymo, studentams praktikos darbų pasirinkimo laiko sąnaudas bei palengvina šiuos darbo procesus. Projektas sukurtas taip, kad galutiniam vartotojų nereikia įdiegti papildomos programinės įrangos, užtenka standartinių operacinės sistemos priemonių (interneto naršyklės), todėl efektyvumas vertinamas kaip aukštas.

4.8.6 Sąsajos paprastumas

Kvalifikuotam vartotojui užtenka perskaityti vartotojo vadovo dokumentą ir apie 3 valandas peržiūrėti sistemą, ir jis bus pilnai susipažinęs su sistema. Pradinių duomenų paruošimas atliekamas tik vieną kartą, o po to lieka tik jais pasinaudoti. Apie galimas klaidas vartotojas yra įspėjamas.

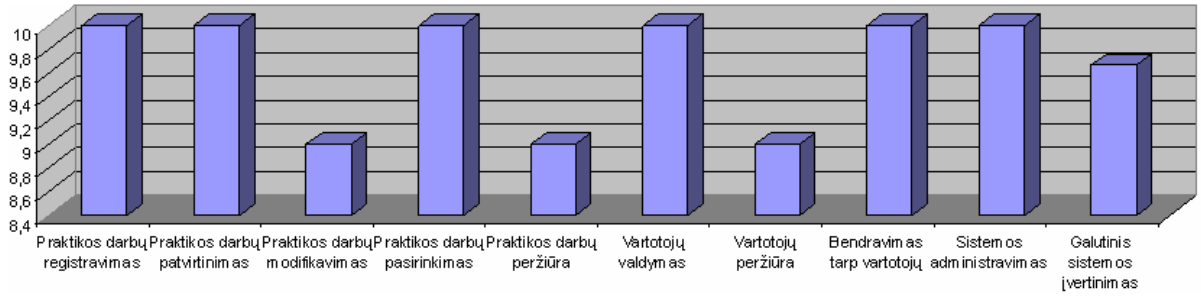
4.8.7 Atitikimas reikalavimų specifikacijai

Norint sulaukti produkto pripažinimo, pirmiausiai sukurtas produktas turi idealiai veikti, atlikti tinkamus darbus tinkamu metu ir be abejonės tenkinti reikalavimų specifikaciją. Kad programinė įranga būtų be klaidų yra labai sunkiai pasiekiama. Didelė kokybės dalis gali būti priskiriama testavimui, tai pasiekti padeda detalus testavimo etapo planavimas. Detaliame testavimo plane pabrėžiama, kokius objektus, duomenis, limitus, ribas ir suvaržymus būtina ištestuoti.

Studentų praktikų ir baigiamųjų darbų portalas turi testinę specifikaciją, kurioje pabrėžiami projekto kūrimo sunkumai, kurie vėliau skaudžiai gali įtakoti projektą. Testavimo plano tikslas yra padėti surasti kuo daugiau klaidų, išsiaiškinti ar programa atitinka reikalavimų specifikaciją, patikrinti ar sistema su skirtingais ir ne visada korektiškais duomenimis elgiasi taip kaip nurodyta techninėje specifikacijoje. Testavimo išvados pateiktos projekto testavimo specifikacijoje.

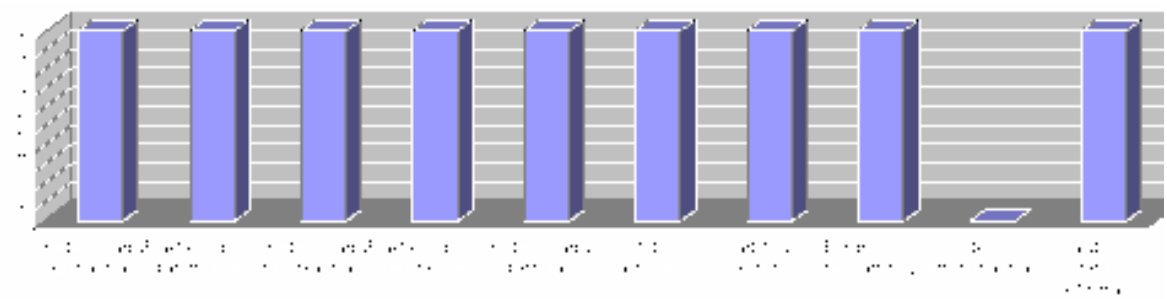
4.8.8 Vertinimo rezultatai

Kokybinis projekto vertinimas pateiktas (41 pav.).



41 pav. Užsakovo projekto įvertinimas

Testavimo rezultatai pateikti (42 pav.).



42 pav. Praėję ir nepraėję testai[1]

4.8.9 Kokybės vertinimo išvados

Projekto programinei įrangai įvertinti buvo naudojami James A. McCall programinės įrangos kokybės įvertinimo faktoriai. McCall programinės įrangos kokybės faktoriai leidžia vertinti programinį produktą tiek iš vartotojo pusės, tiek ir iš sistemos kūrėjo pusės.

Vertinimo rezultatuose matyti, kad projekto užsakovas yra patenkintas galutiniu produktu. Testavime aptiktos klaidos didžiausiame sistemos komponente „Sistemos administravimas“. Rastos klaidos buvo ištaisytos.

[1] Sistemos administravimas ištaisytas

5 ŠABLONŲ SISTEMOS Smarty ir DUOMENŲ ABSTRAKCIJOS KLASĖS PEAR::DB EKSPERIMENTINIS NAŠUMO TYRIMAS

5.1 Eksperimento tikslas ir uždaviniai

Eksperimentinio tyrimo tikslas – patvirtinti arba paneigti šablonų sistemų ir duomenų bazių abstrakcijų naudą sistemų kūrimui. Tyrimai bus atlikti panaudojant sukurto projekto programinius modulius. Svarbiausi uždaviniai, kurių atžvilgiu bus vykdomi tyrimai:

- Pasirinktos (*Smarty*) šablonų sistemos, našumo eksperimentinis tyrimas;
- Ištirti kitų šablonų sistemų našumo savybes ir jas palyginti su pasirinkta (*Smarty*) šablonų sistema;
- PEAR::DB duomenų abstrakcijų klasės eksperimentinis našumo tyrimas.

Visi eksperimentai bus atlikti personalinio kompiuterio pagalba, kurio techniniai parametrai yra šie:

- Operacinė sistema - Windows XP (*Service Pack2*);
- Procesorius - Intel Pentium M processor 1.5 GHZ L2 Cache 2MB;
- Pagrindinė plokštė - Intel i855GM/GME;
- Operatyvioji atmintis – Kingston 512 MB;
- Kietasis diskas – Hitachi 80GB 7200 RPM.

5.2 Smarty šablonų sistemos našumo eksperimentinis tyrimas

Našumo eksperimentinis tyrimas bus atliekamas panaudojant sukurto projekto programinius modulius. Atliksime du našumo eksperimentus. Pirmajame eksperimente panaudosime šablono kintamuosius ir jiems priskirsime reikšmes, antrajame sudarysime kintamo dydžio masyvą ir šio masyvo duomenis perduosim į šabloną, kuriame reikšmes atvaizduosime, panaudodami šablono ciklą.

Pirmoje našumo tyrimo dalyje mes atlikome eksperimentą, šablono kintamųjų kiekį didindami nuo 1 iki 60000 ir jiems priskirdami testines reikšmes matavome skriptų įvykdymo laiką. Eksperimento rezultatai pateikti 12 lentelėje.

12 lentelė. Smarty šablonų sistemos našumo eksperimentas įvertinantis šablono kintamųjų priskyrimą

Eksperimento numeris	Našumo eksperimentas, naudojant Smarty šablonų sistemą be kešavimo ir kompiliavimo funkcijų		Našumo eksperimentas, naudojant Smarty šablonų sistemą su kompiliavimo funkcija, tačiau be kešavimo		Našumo eksperimentas, naudojant Smarty šablonų sistemą su kompiliavimo ir kešavimo funkcijomis	
	Šablono kintamųjų ir jiems priskiriamų reikšmių kiekis	Šablono apdorojimo laikas sekundėmis	Šablono kintamųjų ir jiems priskiriamų reikšmių kiekis	Šablono apdorojimo laikas sekundėmis	Šablono kintamųjų ir jiems priskiriamų reikšmių kiekis	Šablono apdorojimo laikas sekundėmis
1.	1	0,1096271	1	0,0289711	1	0,0287358
2.	10	0,1381399	10	0,0296031	10	0,0294068
3.	100	0,4384808	100	0,0319759	100	0,0311749
4.	1000	3,6318211	1000	0,1613068	1000	0,0544939
5.	5000	19,4253519	5000	0,8694398	5000	0,2628259
6.	10000	38,1758069	10000	1,4530351	10000	0,3526568
7.	20000	80,3070449	20000	2,8704907	20000	0,7344238
8.	30000	122,9827324	30000	4,3142198	30000	1,0954132
9.	40000	162,4471359	40000	5,7156469	40000	1,5309101
10.	50000	209,7107808	50000	7,1282858	50000	1,8792998
11.	60000	251,2051913	60000	8,6464397	60000	2,2554551

Iš rezultatų matyti, kad Smarty šablonų sistema be papildomų funkcijų aktyvavimo veikia labai nenašiai. Įjungus šablonų kompiliavimo funkciją, našumas apytiksliai padidėja 10 kartų. Tai paaiškinama tuo, kad šablonas pakartotinai nebėra interpretuojamas, į šabloną įstatyti kintamieji yra sukompiliuojami į atskirą PHP kalbos failą. Pasikeitus šablonui, kompiliavimas yra atliekamas automatiškai, tačiau kompiliavimo metu našumo rezultatai yra netgi prastesni už Smarty šablonų sistemos su išjungtomis kešavimo ir kompiliavimo funkcijomis. Tai nutinka dėl to, kompiliavimui reikalingas papildomas laikas. Kompiliavimas nesukelia didesnių problemų, kadangi jis atliekamas vieną kartą pasikeitus šablonui. Kešavimo panaudojimas gali papildomai pagerinti našumo parametrus. Tačiau kešavimo panaudojimas dažnai yra komplikuotas, nes nustatytą laiką (pagal nutylėjimą vieną valandą) šablono kintamieji panaudos senus duomenis. Kešavimas pasiteisina tuose vietose, kada duomenys yra retai kintantys bei tas pats šablonas yra dažnai naudojamas.

Antroje našumo tyrimo dalyje mes atlikome eksperimentą, užpildėme kintamo dydžio masyvo nuo 1 iki 60000 elementų, testiniais duomenimis, perdavėme šiuo masyvo duomenis į šabloną ir šablono ciklo pagalba matavome masyvo duomenų išvedimo laiką (šablono ciklo apdorojimo laiko matavimas). Eksperimento rezultatai pateikti 13 lentelėje.

13 lentelė. Smarty šablonų sistemos našumo eksperimentas įvertinantis masyvo reikšmių išvedimą, šablono ciklo pagalba

Eksperimento numeris	Našumo eksperimentas, naudojant Smarty šablonų sistemą be kešavimo ir kompiliavimo funkcijų		Našumo eksperimentas, naudojant Smarty šablonų sistemą su kompiliavimo funkcija, tačiau be kešavimo		Našumo eksperimentas, naudojant Smarty šablonų sistemą su kompiliavimo ir kešavimo funkcijomis	
	Šablono ciklo iteracijų kiekis	Šablono apdorojimo laikas sekundėmis	Šablono ciklo iteracijų kiekis	Šablono apdorojimo laikas sekundėmis	Šablono ciklo iteracijų kiekis	Šablono apdorojimo laikas sekundėmis
1.	1	0,1252739	1	0,0297429	1	0,0287358
2.	10	0,1270579	10	0,0312640	10	0,0294068
3.	100	0,1355221	100	0,0348799	100	0,0311749
4.	1000	0,3013093	1000	0,0792579	1000	0,0544939
5.	5000	0,9884215	5000	0,5994319	5000	0,2628259
6.	10000	2,0963959	10000	1,5352118	10000	0,3526568
7.	20000	4,9918768	20000	3,3428328	20000	0,7344238
8.	30000	8,1221652	30000	7,2603704	30000	1,0954132
9.	40000	11,4377648	40000	10,5340621	40000	1,5309101
10.	50000	15,0430312	50000	14,6987359	50000	1,8792998
11.	60000	19,2821889	60000	18,2552971	60000	2,2554551

Šis eksperimentas taip pat patvirtina, kad geriausios Smarty šablonų sistemos našumo charakteristikos yra pasiekiamos naudojant kompiliavimo ir kešavimo funkcijas.

5.3 Eksperimentinis šablonų sistemų našumo palyginimas

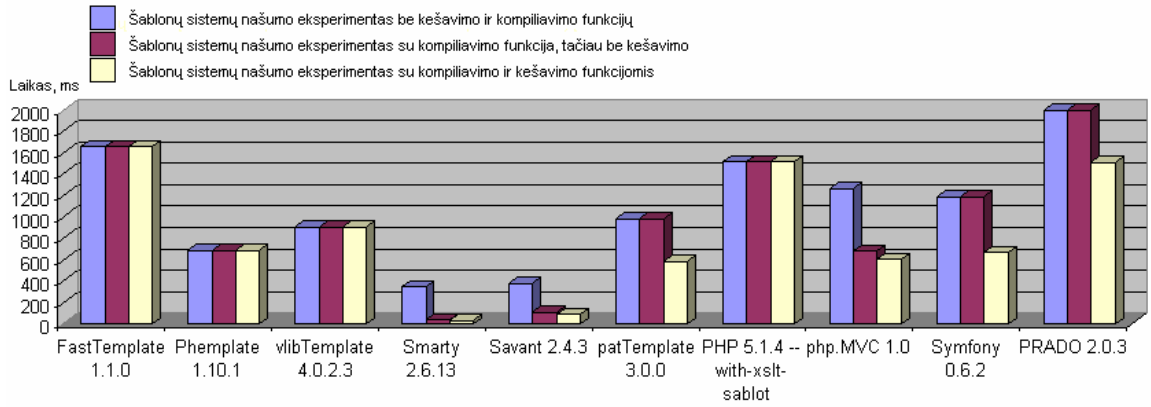
Atliktuose eksperimentiniuose našumo matavimuos, matyti tik Smarty šablonų sistemos rezultatai. Išanalizavę šiuos rezultatus galime išskirti tik gerąsias ar blogąsias Smarty šablonų sistemos savybės, tačiau norint įvertinti šią šablonų sistemą kitų šablonų sistemų kontekste yra būtina atlikti kitų šablonų sistemų eksperimentinius našumo matavimus ir juos palyginti su Smarty šablonų sistemos našumo eksperimentų rezultatais. Analizuojant šablonų sistemas išskyrėme šablonų sistemų grupes, todėl 14 lentelėje pateikiame šių šablonų sistemų našumo eksperimentus, panaudojant šablono kintamųjų priskyrimą.

14 lentelė. Šablonų sistemų našumo eksperimentas įvertinantis šablono kintamųjų priskyrimą

Eksperimento numeris	Šablonų sistemų našumo eksperimentas be kešavimo ir kompiliavimo funkcijų		Šablonų sistemų našumo eksperimentas su kompiliavimo funkcija, tačiau be kešavimo		Šablonų sistemų našumo eksperimentas su kompiliavimo ir kešavimo funkcijomis	
	Šablono kintamųjų ir jiems priskiriamų reikšmių kiekis	Šablono apdorojimo laikas sekundėmis	Šablono kintamųjų ir jiems priskiriamų reikšmių kiekis	Šablono apdorojimo laikas sekundėmis	Šablono kintamųjų ir jiems priskiriamų reikšmių kiekis	Šablono apdorojimo laikas sekundėmis
Paprastos šablonų sistemos FastTemplate 1.1.0						
1.	20	1,074312	nepalaikoma	-	nepalaikoma	-
2.	100	2,236323	nepalaikoma	-	nepalaikoma	-
Phemplate 1.10.1						
3.	20	0,233212	nepalaikoma	-	nepalaikoma	-
4.	100	1,133226	nepalaikoma	-	nepalaikoma	-
vlibTemplate 4.0.2.3						

5.	20	0,347611	nepalaikoma	-	nepalaikoma	-
6.	100	1,465677	nepalaikoma	-	nepalaikoma	-
Šablonų sistemos, palaikančios kompiliavimą ir papildomus modulius						
Smarty 2.6.13						
7.	20	0,178972	20	0,034631	20	0,030838
8.	100	0,512902	100	0,041198	100	0,041198
Savant 2.4.3						
9.	20	0,143941	20	0,091349	20	0,071349
10.	100	0,612387	100	0,127487	100	0,099372
patTemplate 3.0.0						
11.	20	0,644324	nepalaikoma	-	20	0,060323
12.	100	1,311529	nepalaikoma	-	100	1,101232
XSLT transformacijomis paremta šablonų sistema						
PHP 5.1.4 --with-xslt-sablot						
13.	20	1,121152	nepalaikoma	-	nepalaikoma	-
14.	100	1,912344	nepalaikoma	-	nepalaikoma	-
Modelis – Vaizdas – Kontroleris struktūrinės šablonų sistemos						
php.MVC 1.0						
15.	20	0,812171	20	0,212171	20	0,103710
16.	100	1,713455	100	1,142343	100	1,102122
Symfony 0.6.2						
17.	20	0,721312	nepalaikoma	-	20	0,225434
18.	100	1,654566	nepalaikoma	-	100	1,112138
PRADO 2.0.3						
19.	20	1,744912	nepalaikoma	-	20	1,123434
20.	100	2,244354	nepalaikoma	-	100	1,889343

Išanalizavus 14 lentelės duomenis aiškiai matyti, kad Smarty šablonų sistemą yra našiausia lyginant su kitomis šablonų sistemomis. Eksperimentų metu nustatyta, kad papildomų šablonų funkcijų panaudojimas (kompiliavimas, kešavimas) žymiai padidina internetinės aplikacijos našumo charakteristikas. 3 lentelės rezultatų apibendrinimą pateikiame (43 pav.).



43 pav. Apibendrintas šablonų sistemų našumo eksperimentas

5.4 PEAR::DB duomenų abstrakcijų klasės eksperimentinis našumo tyrimas

PEAR::DB duomenų abstrakcijos klasės eksperimentinio tyrimo tikslas parodyti našumo skirtumus tarp gimtųjų PHP programavimo kalbos funkcijų, skirtų darbui su DBVS ir objektiškai orientuotos PEAR:DB. Eksperimentą atliksime pasinaudodami populiariausiomis

atviro kodo duomenų bazių valdymo sistemomis MySQL ir PostgreSQL bei pasinaudosime sukurto projekto duomenų baze. Našumo eksperimentuose bus matuojamas SQL komandų atlikimo greitis panaudojant tiek gimtąsias PHP funkcijas, tiek objektiškai orientuotą PEAR::DB. Atliekant eksperimentą students lentelę užpildysime testiniais duomenimis. Prieš atliekant eksperimentus students lentelė bus užpildyta 10000 įrašų – testinių duomenų. Eksperimentų rezultatus pateikiame 15 lentelėje.

15 lentelė. PEAR:DB ir gimtųjų PHP funkcijų

DBVS / SQL	MySQL 5.0		PostgreSQL 8.1	
	Panaudojant gimtąsias PHP funkcijas	**PEAR::DB**	**Panaudojant gimtąsias PHP funkcijas**	**PEAR::DB**
SELECT	0.33213 sec	0.33833 sec	0.64123 sec	0.64523 sec
UPDATE	1.00432 sec	1.01233 sec	0.55653 sec	0.55981 sec
DELETE	0.55133 sec	0.55434 sec	0.45134 sec	0.45721 sec

Eksperimentų rezultatuose matyti, kad PEAR::DB yra lėtesnė už gimtąsias PHP prieigos prie duomenų bazių funkcijas. Tai paaiškinama tuo, kad PEAR::DB yra gimtųjų PHP prieigos prie duomenų bazių funkcijų apvalkalas, kuris palengvina programuotojų darbą, tačiau prideda papildomą sudėtingumo lygmenį. Šios sąsajos pridėjimas, kaip tik ir įtakoja šiuos greičio skirtumus.

Abejojant ar vertą naudoti PEAR::DB dėl to, kad ji yra lėtesnė už gimtąsias PHP prieigos prie duomenų bazių funkcijas, verta atkreipti dėmesį į tai, kad objektiškai orientuota, unifikuota PEAR::DB žymiai paspartiną ir palengviną internetinių aplikacijų kūrimą, ateityje yra išvengiamos portatyvumo problemos.

6 IŠVADOS

1. Šiame darbe pateikiama egzistuojančių šablonų sistemų taksonomija, pagal pateiktą taksonomiją atlikta šablonų sistemų analizė, palyginimas, įvertinamas tinkamumas šiandienos interneto aplikacijų kūrimui.
2. Detaliai aprašyti žiniatinklio aplikacijų kūrimo modeliai, modelių veikimo principai, kurie detalizuojami grafinėmis iliustracijomis.
3. Pasiūlytas ir smulkiai apibūdintas internetinių aplinkų kūrimo modelis, kuriame panaudojama šablonų sistema Smarty ir duomenų abstrakcijų klasė PEAR::DB. Modelyje aprašoma kaip reikia kurti žiniatinklio aplikacijas, jų kūrimą suskirstant į modulius atsakingus už prezentaciją, verslo taisykles ir duomenis. Pateikiamas šio modelio trijų sluoksnių surišimo iliustravimas ir atlikimo eiliškumas.
4. Panaudodami trijų sluoksnių siūlomą modelį atlikome projektą „Studentų praktikos darbų portalas“. Sėkmingai realizuotas portalas, kuris leidžia automatizuoti praktikos darbų pasirinkimą, pasiūlymą ir atlikti praktikos darbų valdymą yra įdiegtas KTU informacijos sistemų katedros serveryje.
5. Panaudojus sukurto portalo programinius modulius ištyrėme šablonų sistemos Smarty ir duomenų abstrakcijos klasės PEAR:DB našumo charakteristikas. Našumo charakteristikos buvo palygintos su kitomis šablonų sistemomis. Atlikus našumo eksperimentus aiškiai matyti, kad šablonų sistemos, kurios palaiko papildomas funkcijas (kešavimas, kompiliavimas) yra žymiai našesnės. Norint našesnio šablonų sistemos veikimo būtina aktyvuoti kompiliavimo bei kešavimo funkcijas. Eksperimentiniuose našumo tyrimuose pastebėta, kad Smarty šablonų sistemos našumo charakteristikos yra geriausios, todėl rekomenduotina šią šablonų sistemą naudoti didelio apkrovimo sistemose, ten kur didelis našumas yra būtinas. Duomenų abstrakcijų klasės PEAR::DB našumo rezultatai gauti blogesni lyginant su gimtosiomis PHP kalbos prieigos prie duomenų bazių funkcijomis. Tačiau pastebėta, kad šis skirtumas apytiksliai yra vienodas ir nepriklausomas nuo duomenų dydžio. Todėl rekomenduotina šią duomenų abstrakcijos klasę naudoti tuose žiniatinklio aplikacijose, kuriuose yra labai svarbios portatyvumo savybės.
6. Darbo tematika buvo publikuotas straipsnis XI-oje tarpuniversitetinės doktorantų ir magistrantų konferencijos “Informacinės technologijos 2006” pranešimų medžiagos leidinyje.

7 LITERATŪRA

1. LAURENCE, M. Rapid Template-Driven Web Development. *The Tenth International World Wide Web Conference.* Hong Kong, 2001.

2. SIRIN, E.; PARSIA, B.; HENDLER, J. Template-based Composition of Semantic Web Services. *Maryland Information and Network Dynamics Lab Semantic Web Agents Project*. University of Maryland, USA, 2005.

3. CRUZ, L. Y. Three-Tier Development with PHP 5. Iš *O'REILLY ONlamp.com PHP DEVCENTER* [interaktyvus]. 2005, rugsėjis [žiūrėta 2006-01-04]. Prieiga per internetą: http://www.onlamp.com/pub/a/php/2004/12/09/three_tier.html.

4. MAIA, J. P. Pear::DB Primer. Iš *O'REILLY ONlamp.com PHP DEVCENTER* [interaktyvus]. 2005, rugsėjis [žiūrėta 2006-02-09]. Prieiga per internetą: http://www.onlamp.com/lpt/a/1356.

5. CHRISTOPHER, R. Manage Your Content With PHP. *The SemanticWeb - ISWC 2004, 4th International SemanticWeb Conference*. Galway, Ireland, 2005.

6. MULYE, R. A Semantic Template Based Designer for Web Processes. *IEEE International Conference on Web Services*. Florida, USA, 2005.

7. ARPINAR, I.; Zhang, R.; Aleman-Meza B.; Maduko A. Ontology-Driven Web Services Composition Platform. *2004 IEEE International Conference on E-Commerce Technology (CEC'04)*. California, USA, 2004, p. 146-152.

8. KETTLER, B.; STARZ, J.; MILLER, W., HAGLICH, P. A Template-Based Markup Tool for Semantic Web Content. *The SemanticWeb - ISWC 2005, 4th International SemanticWeb Conference*. Galway, Ireland, 2005, p. 446-460.

9. Lozier, B. Template Engines [interaktyvus]. 2003, vasaris [žiūrėta 2006-02-24]. Prieiga per internetą: http://www.massassi.com/php/articles/template_engines.

10. SOLIN, D. Modular PHP Development with FastTemplate. Iš *O'REILLY ONlamp.com PHP DEVCENTER* [interaktyvus]. 2003, vasaris [žiūrėta 2006-02-21]. Prieiga per internetą: http://www.onlamp.com/pub/a/php/2003/10/02/modular_php.html?page=1.

11. Šalna, J. Phemplate šablonų sistemos panaudojimas. *Lietuvos PHP konferencija 2004*. Vilnius, 2004.

12. Lozier, B. Beyond The Template Engine. Iš sitepoint [interaktyvus]. 2003, rugsėjis [žiūrėta 2006-03-01]. Prieiga per internetą: http://www.sitepoint.com/print/beyond-template-engine.

13. CRESCENZI, V.; MERIALDO, P.; MISSIER, P. Clustering Web pages based on their structure. *Data & Knowledge Engineering, Volume 54*.2005, Nr. 3, p. 279-299 p.

14. GANGEMI, G. Ontology Design Patterns for Semantic Web Content. *The SemanticWeb - ISWC 2005, 4th International SemanticWeb Conference*. Galway, Ireland, 2005, p. 262-276 p.

15. MIKHAILIAN, A. Occasional XSLT for Experienced Software Developers. Iš DevX [interaktyvus]. 2005, liepa [žiūrėta 2006-02-10]. Prieiga per internetą: http://www.devx.com/xml/Article/28610/1954?pf=true.

16. ZHIMA, G.; MIN, L.; XIAOLING, W. Scalable XSLT Evaluation. *Advanced Web Technologies and Applications: 6th Asia-Pacific Web Conference.* Hangzhou, China, 2004, p. 190 – 200.

17. FITZGERALD, M. *Learning XSLT*. United States of America, 2004.

18. VION, J.; LUX, V.; PIETRIGA, E. Experimenting with the Circus Language for XML Modeling and Transformation. *Proceedings of the 2002 ACM symposium on Document engineering*. Virginia, USA, 2002, p. 82-87.

19. GILMORE, W. J. PHP and the Sablotron Processor. *Oreilly OnLamp Conference*. Sebastopol, USA 2001.

20. REENSKAUG, T. The Model-View-Controller (MVC) Its Past and Present Trygve Reenskaug. *JAOO Conference*. Aarhus, Denmark, 2003.

21. BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; STAL, M. Pattern - Oriented Software Architecture. New York, 2000.

22. GRESH, J. E. The Collection Switch Design Pattern. *Rensselaer Computer Science Conference*. Oslo, 2004.

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Terminas	Aprašymas
Apache	Web servisas.
API	Programavimo sąsaja (*Application Programming Interface*).
DB	Duomenų bazė.
DBVS	Duomenų bazių valdymo sistema.
FireFox	Interneto naršyklė.
FreeBSD	Operacinė sistema.
GNU/GPL	Licencijos tipas.
IDE	(*Integrated Development Environment*) – Integruota projektavimo aplinka. Paprastai būna specializuota tam tikrai kalbai. Apima programų teksto redaktorių, konkrečios kalbos kompiliatorių, programų derinimo priemones ir pagalbos sistemą. Gali turėti specialias priemones vartotojo aplinkos projektavimui.
Interneto naršyklė	Programa, skirta peržiūrinėti Interneto tiekiamus.
IS	Informacinė sistema.
LiemSIS	Lietuvos mokslo ir studijų informacijos sistema**.**
Linux	Operacinių sistemų šeima.
Mac	Operacinių sistemų šeima.
Objektas	Kuriamos sistemos dalis.
PEAR::DB	Duomenų abstrakcijos klasės pavadinimas.
PHP	Web puslapių programavimo kalba.
phpPgAdmin	PostgreSQL duomenų bazių valdymo įrankis, veikiantis Web serviso pagalba.
Portalas	Vartai į informaciją, žinių bazę, leidžiantys vartotojams kartu dalintis ar prieiti prie jiems reikalingos informacijos, laikomos skirtingose sistemose. Daugelis portalų siūlo integruotą dokumentų valdymo sistemą ir įrankius, pagerinančius bendruomenės komunikavimą: forumus, kalendorius, projekto informacijos segregaciją, svetainės personalizaciją
PostgreSQL	Duomenų bazių valdymo sistema.
RUP	Rational unifikuotas procesas (*Rational Unified Process*).
Smarty	Šablonų variklis, skirtas PHP.
SQL	Struktūrizuotų užklausų kalba (Structured Query Language).
UML	Unifikuota modeliavimo kalba (*Unified Modelling Language*).
Unix	Operacinių sistemų šeima.
WEB	Pasaulio žiniatinklis.
Windows	Operacinių sistemų šeima.
XHML	Hiperteksto žymių kalba.
XML	Duomenų objektų aprašymo kalba.

Template Based and Data Abstraction Driven Web Applications Development Model

SUMMARY

Web pages written with simple web development technologies were either written in code (such as C with CGI, or Java with Servlets) which contained lots of HTML-writing code snippets, or written with numerous code segments embedded in HTML (such as PHP or ASP). This led to problems as increasingly sophisticated Web page designs got harder and harder to 'activate' with data, and become increasingly brittle when dealing with change requests. Complex Web pages development forces to use new technologies that helps in development phases. It is possible to separate business logic from presentation using template engines. It's not easy to choose a proper template engine system. We classified existing template systems. By this way analysis were accomplished.

Using web template system, which supports compilation and caching features combining with MVC element data abstraction, we proposed web applications development model. In this work our proposed model has been applied for the development of students practice jobs - web portal. This portal was realized using PHP technologies which includes Smarty template engine, data abstraction class PEAR::DB and PHP 5.0 programming language.

We research our proposed web development model performance characteristics. Performance experiments were done with Smarty template engine and PEAR::DB data abstraction class.

PRIEDAI

1 priedas. Konferencijos straipsnis

RIMKUS, M. WEB Šablonų taksonomija.*Informacinės technologijos 2006. 11 – oji tarpuniversitetinė doktorantų ir magistrantų konferencija. Konferencijos pranešimų medžiaga. I dalis*. Kaunas, 2006, p. 194-199. ISBN 9986-19-877-1

WEB ŠABLONŲ SISTEMŲ TAKSONOMIJA

Mindaugas Rimkus

Kauno technologijos universitetas

Sudėtingas Web puslapių kūrimas skatina taikyti technologijas, palengvinančias kūrimo procesą. Panaudojant šablonų sistemas galima atskirti Web aplikacijos biznio logiką nuo atvaizdavimo. Nėra paprasta išsirinkti tinkamą sistemą, kuri padėtų įgyvendinti tikslus. Aš pateiksiu egzistuojančių šablonų sistemų klasifikaciją ir kiekvienos klasifikacinės grupės analizę bei vertinimus.

1. Įvadas

Web aplikacijų kūrimas praėjo ilgą raidos kelią, pradedant nuo HTML atsiradimo ir puslapių kodavimo be papildomų įrankių. Pirmosios Web technologijos buvo nepajėgios generuoti puslapius pagal vartotojo pasirenkamas nuostatas. Duomenų bazėmis paremti tinklapiai kuriami nuskaitant duomenis bei juos panaudojus statiniai puslapiai sujungiami į navigacinę sistemą [1]. Pasikeitus informacijai duomenų bazėje, tuomet reikėdavo visą Web aplikaciją sugeneruoti iš naujo.

Plėtojantis serverio tipo programinei įrangai, atsirado technologijos CGI, ASP, PHP ir daug kitų. Išankstinis Web puslapių generavimas tapo technologine relikvija. Galimybė puslapius generuoti pagal konkrečią užklausą, įgalino sukurti pilnavertę Web aplikaciją, kuri sąveikauja su vartotoju. Iš duomenų bazės lengvai galima išgauti duomenis pagal tam tikrą kontekstą ir sukurti HTML atsakymą.

Web aplikacijų generavimas pagal kontekstą panaudojant naujas Web technologijas iškėlė kitą problemą. Puslapiai sukurti kalbomis C, CGI, Java ir kt. talpina didžiulius HTML kodo įterpinius arba HTML dokumente PHP, ASP ir kt. programinio kodo segmentus. Šis programinio kodo ir HTML maišymas įtakoja vis didėjantį Web puslapių projektavimo, kodavimo sudėtingumą. Tokių puslapių rekonstravimas tampa sudėtinga rutina, kuri užima didelę laiko dalį.

Verslo logikos ir atvaizdavimo atskyrimo problema šiandien išsprendžiama šablonų pagalba. Naudojant šablonus, puslapio dizaineris gali kurti puslapio grafinį vaizdą įterpdamas kintamojo žymę, kurią programuotojas panaudoja aktyvuodamas puslapį. Pastaruoju metu šablonų sistemos yra labai populiarios ir naudojamos beveik visuose sudėtingesniuose projektuose.

Aš pateiksiu atviro kodo PHP programavimo kalbai skirtų šablonų sistemų taksonomiją, analizę ir vertinimą.

2. Web šablonų taksonomija ir analizė

Kadangi egzistuoja daug įvairių Web šablonų sistemų, visas išnagrinėti būtų nepakeliama užduotis, todėl egzistuojančias šablonų sistemas suskirstysiu į grupes. Pirmai grupei priskirsiu paprasčiausias šablonų sistemas, tiksliau programavimo stilių. Antrai kategorijai - trečių šalių sukurtas šablonų sistemas, kuriose realizuotas tik paprasčiausias šablonų kintamųjų pakeitimas tam tikra informacija, kurią nustato programuotojas. Dizaineris yra atsakingas už tekstinio šablono sukūrimą ir kintamųjų pažymėjimą. Trečios kategorijos šablonų sistemos turi būti sudarytos iš visų savybių, kurias turi antros kategorijos šablonų sistemos, pridedant reikalavimus šablonų kompiliavimui bei papildomų modulių palaikymui. Į ketvirtą kategoriją įtrauksiu XML transformacijų pagrindu veikiančias šablonų sistemas, kurios įgyvendinamos XSLT pagalba. Paskutinei grupei priskirsiu struktūrines šablonų sistemas, kurios sudarytos iš trijų pagrindinių komponentų modelio, vaizdo ir kontrolerio. Modelis - atsakingas už priėjimą prie duomenų. Vaizdas – Web puslapio grafinė pusė, dizainas. Kontroleris susieja modelį ir vaizdą, suformuluoja atsakymą vartotojui.

Apibendrinus galima išskirti tokias šablonų sistemų grupes:

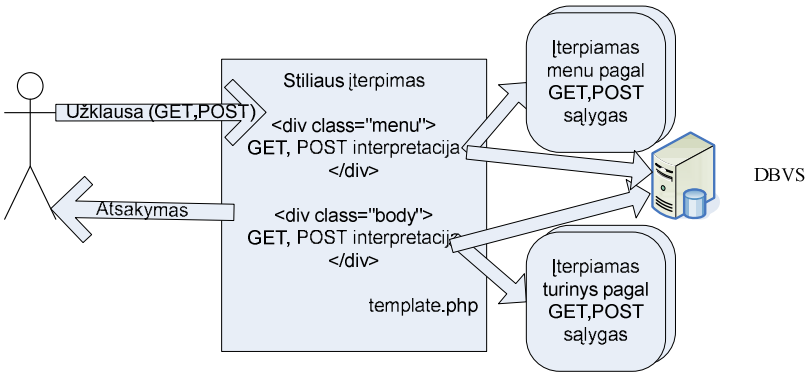
6. Standartinėmis priemonėmis paremtas šablonų įgyvendinimas;
7. Paprastos šablonų sistemos;
8. Šablonų sistemos palaikančios kompiliavimą ir papildomus modulius;
9. XSLT transformacijomis paremta šablonų sistema;
10. Modelis – Vaizdas – Kontroleris struktūrinės šablonų sistemos.

2.1 Standartinėmis priemonėmis paremtas šablonų įgyvendinimas

Standartinėmis priemonėmis paremtos šablonų sistemos yra labai paprastos savo funkcionalumu ir netenkina logikos ir dizaino atskyrimo reikalavimų. Aptarsiu siūlomus metodus, kaip standartinėmis priemonėmis atskirti puslapio stilių, turinį ir struktūrą PHP programavimo kalboje.

Tradiciškai stiliaus atskyrimui nuo turinio naudojamos CSS kaskadinių stilių lentelės ir XHTML praplėsta hiperteksto žymėjimo kalba, kuri suderinama su XML. CSS dėka galima lengvai atnaujinti puslapio stilių ar sukurti keletą stilių. CSS formavimui panaudojus PHP galima žengti pirmą žingsnį link modulinės sistemos, kurios pagalba bus lengviau atnaujinamas ne tik stilius, bet ir struktūra [2]. Iš esmės tai jau yra labai riboto pobūdžio turinio valdymo sistemą.

Pagrindinis šablonas bus sudarytas iš XHTML, kuris skirtas struktūriniam žymėjimui, CSS atsakingas už turinio stilių ir PHP atliks šio proceso valdymą 1 pav.



1 pav. Standartinėmis priemonėmis paremta šablonų sistemos realizacija

Pasinaudojant GET, POST metodais ir PHP pagalba galima panaudoti kelias stilių lenteles bei importuoti skirtingus meniu ar turinio dokumentus tame pačiame šablone (mano atveju template.php).

Tai pats lengviausias būdas suskirstyti kuriamą Web puslapį į tam tikrus modulius ir lengvai atlikti tų modulių apjungimą bei panaudojimą. Pagrindiniai metodikos trūkumai yra negalėjimas atskirti PHP ir HTML kodo, lėtas veikimas.

2.2 Paprastos šablonų sistemos

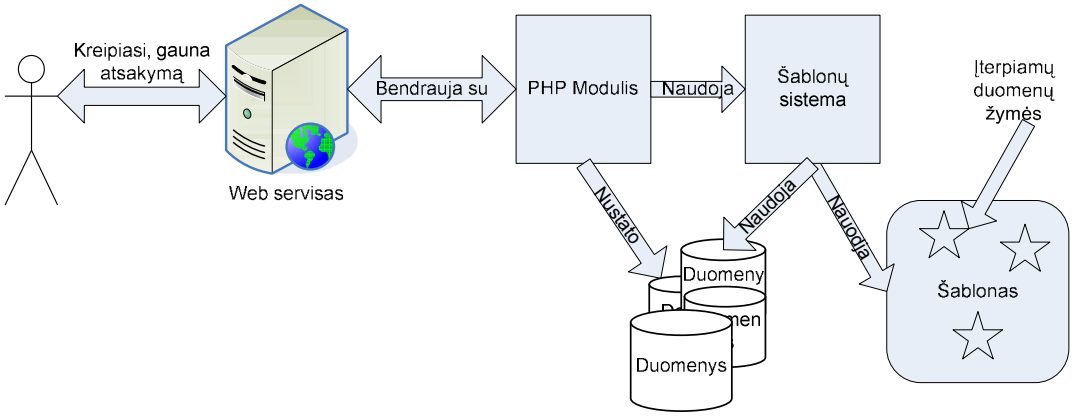
Paprastos šablonų sistemos yra programinė įranga, kuri apdoroja įeinantį tekstą (šabloną) ir pateikia išeinantį tekstą (atsakymą) 2 pav. Paprastų šablonų sistemų tikslas yra atskirti HTML kodą nuo PHP, jų nemaišyti kartu. Sudėtingos šablonų sistemos nuo paprastų sistemų skiriasi tuo, kad jų tikslas yra atskirti logiką nuo atvaizdavimo bei suteikti papildomų savybių: kešavimas (*Caching*), įskiepių palaikymas, suderinamumas su XML, saugumo maksimizavimas, greičio praradimo minimizavimas. Visų šablonų sistemų tikslas nėra pašalinti visą logiką iš HTML. Šablonų sistemos turi atskirti prezentacinę logiką nuo biznio logikos.

Paprastos šablonų sistemos paprasčiausiai atskiria serverinės pusės kodą nuo klientinės pusės (PHP ir HTML atskyrimas). Šablonų sistemos išsprendžia šias dvi pagrindines problemas:

3. Kaip įgyvendinti prezentacinės logikos atskyrimą nuo biznio logikos;
4. Kaip atskirti sudėtingą PHP kodą nuo HTML.

Šios metodikos idėja yra įgalinti HTML puslapių dizainerius neturinčius PHP kodavimo patirties pakeisti puslapių dizainą, neatsižvelgiant į PHP kodą. PHP programuotojui turi būti sudaryta galimybė visą dėmesį skirti tik į kodavimą, nekreipiant dėmesio į dizaino elementus.

Šablonų sistemų atsiradimas taip pat įnešė papildomą sudėtingumo lygmenį Web aplikacijų kūrimo procese. Pirmiausiai, kelis kartus padidėjo projektą sudarančių failų skaičius. Dažniausiai vienas PHP failas yra atsakingas už biznio logiką, kitas už dizaino struktūrą bei vidinis turinio šablonas, kuris įstatomas į pagrindinį šabloną. Prie šių failų prisideda ir šablonų sistemos klasių, jų realizacijų failai.



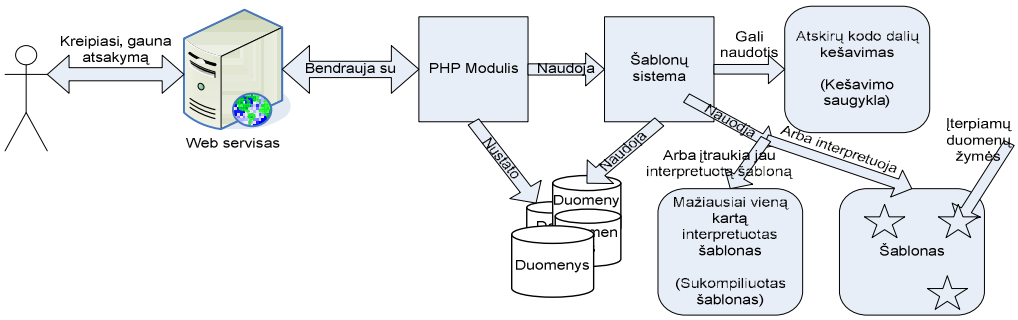
2 pav. Paprastos šablonų sistemos realizacija

Panaudojant šablonų sistemas skriptų vykdymas taip pat tampa sudėtingesnis. Šablono failai turi būti ne tik įterpiami, bet jie privalo būti gramatiškai teisingi. Taigi tai yra svarbus greičio praradimo momentas, paprastos šablonų sistemos neturi papildomų greitį spartinančių priemonių kaip kešavimas ar kompiliavimas. Šablonų sistema yra pseudo kalba PHP skriptų interpretatoriui.

Ši metodologija tinkama mažo ir vidutinio dydžio projektams, kuriems nereikalingos įmantresnės šablonų savybės kaip kešavimas, papildomų modulių palaikymas, griežtas suderinamus su XML. Tai plačiausiai paplitusi metodika tarp Web aplikacijų kūrėjų.

2.3 Šablonų sistemos palaikančios kompiliavimą ir papildomus modulius

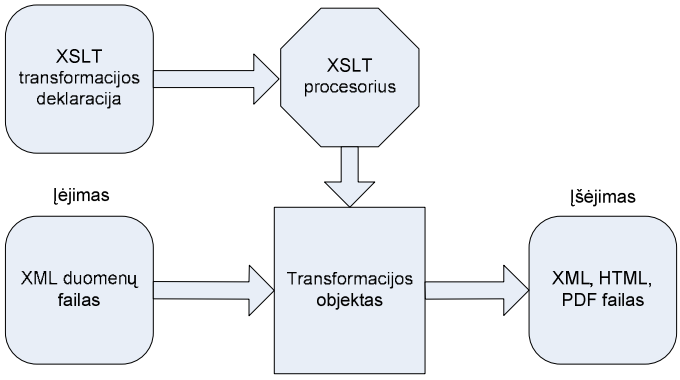
Dideliuose projektuose paprastų šablonų sistemų panaudojimas yra mažai tikėtinas, nes jie susideda iš tūkstančių failų ir visų tų failų integravimas į vieną visumą iškelia greičio ir saugumo problemas. Norint maksimaliai sumažinti šablonų interpretavimo laiką, buvo sugalvotas šablonų kompiliavimas ir kešavimas. Sudėtingų šablonų metodika remiasi paprastų šablonų ideologija, tik žymiai sugriežtina idėja, kad ne tik HTML turi būti atskirtas nuo PHP, bet ir biznio logika turi būti nepriklausoma nuo atvaizdavimo [3]. Sudėtingų šablonų sistemų schema pateikiu 3 pav.



3 pav. Sudėtingos šablonų sistemos prototipas

2.4 XSLT transformacijomis paremta šablonų sistema

XSL transformacijos arba XSLT yra XML pagrindu sukurta kalba naudojama transformuoti XML dokumentus. Transformuojamas XML dokumentas nėra keičiamas, jis yra tik įėjimas XSLT transformacijų sistemai, pagal šio dokumento turinį yra sukuriamas naujas transformuotas XML dokumentas 4 pav. XML dokumentas apdorotas XSLT procesoriaus gali būti išvedamas standartine XML sintakse arba kitais populiariais formatais: HTML, tiesiog tekstu ar pdf dokumentu [4]. Dažniausiai XSLT yra naudojamas norint transformuoti XML duomenų failą į Web puslapius ar PDF dokumentus. XML duomenų failo transformavimas į Web puslapius kaip tik yra viena iš šablonų sistemos realizacijų. Transformacijomis paremta šablonų sistema nuo anksčiau aptartų šablonų skiriasi tuo, kad šablone pažymėtų žymių įstatymo kintamaisiais požiūris pakeičiamas į XML duomenų failo transformavimą į norimą duomenų atvaizdavimo formą (*HTML, PDF*).



4 pav. XSLT modelis

Vienas iš XSLT privalumų - galimybė apsirašyti kelias transformacijas (atvaizdavimus) vienam duomenų failui. Turint vieną XML failą galima lengvai sukurti skirtingai atrodančius Web puslapius.

Kadangi XSLT yra standartas, o ne papildomas klasių rinkinys vienai ar kitai programavimo kalbai, pasinaudoti šia metodika galima beveik visuose programavimo sistemose. Projektai, kuriuose duomenų mainai atliekami XML pagalba, XSLT yra pats patogiausias būdas šiuos XML dokumentus transformuoti į XHTML ar PDF.

Viena problematiškiausių vietų panaudojant XSLT transformacijas PHP kalboje – suderinamumo stoka. Norint įdiegti XSLT į PHP reikalaujama papildomų bibliotekų įdiegimo bei PHP perkompiliavimo. Dažnai realizuotas projektas įgyvendinantis XSLT transformacijas būna lėtesnis už šablonų sistemas su kompiliavimo palaikymu. Išlieka komplikuota dizainerio darbo specifika. Web puslapių dizaineris turi gerai išmanyti XHTML,

XSL, WML. Naudojant šią metodiką išauga reikalavimai dizainerių klasifikacijai, o tai neigiamai veikia projekto biudžetą.

2.5 Modelis – Vaizdas – Kontroleris struktūrinės šablonų sistemos.

Modelis – Vaizdas – Kontroleris (*MVC*) yra programinės įrangos architektūra, kuri aprašo kaip suskaidyti aplikacijos duomenų modelį, vartotojo sąsają ir kontrolės logiką į tris skirtingus komponentus [5]. Vieno komponento modifikavimas turi minimaliai įtakoti kitus komponentus, idealiausiai jų visiškai neįtakoti.

MVC dažnai yra vadinamas programinės įrangos kūrimo šablonas. Tačiau MVC daugiau apima architektūrinę dalį negu tipinį kūrimo šablono apibūdinimą. Teisingiausias MVC terminas yra pasiūlytas F. Buschmann [6] architektūrinis šablonas arba agregatinis kūrimo šablonas.

2.5.1 Modelis

Modelis yra specifinės srities informacijos reprezentacija, funkcionalumo šerdis, kurioje programinė įranga veikia. MVC tikslas yra atskirti modelį nuo vaizdo ir kontrolerio, kurie kartu suformuoja programinės įrangos grafinę sąsają.

Modelis reprezentuoja bizinio duomenis ir taisykles, kurios valdo priėjimą ir keitimą šių duomenų. Dažniausiai modelis panaudojamas kaip programinės įrangos aproksimacija realaus gyvenimo procesui. Realaus gyvenimo modeliavimo technikos pritaikomos apibrėžiant modelį.

2.5.2 Vaizdas

Vaizdas yra skirtas modelio suformuotų duomenų atvaizdavimui galutiniam vartotojui bei turi laisvą priėjimą prie modelio, bet jam draudžiama pakeisti modelio būseną. Vaizdai yra tik skaitymo modelio būsenos reprezentacija. Duomenų skaitymas iš modelio atliekamas standartiniais modelyje aprašytais metodais.

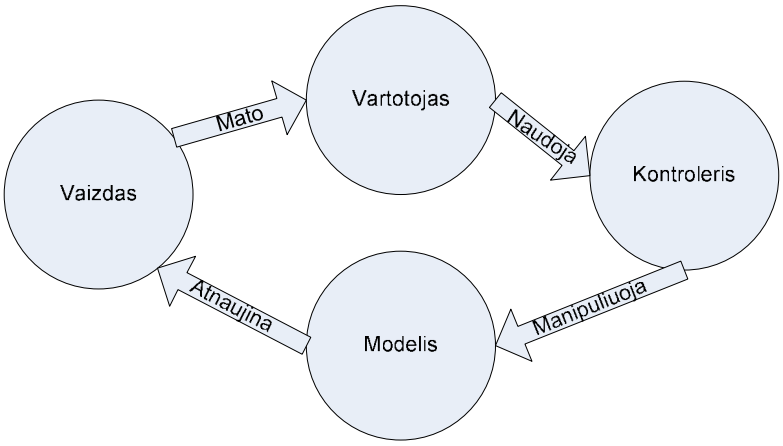
2.5.3 Kontroleris

Kontroleris priima ir transliuoja įėjimą į užklausas, kurios skirtos modeliui arba vaizdui. Paprastai kontroleris yra atsakingas už modelyje esančių metodų iškvietimą, kurie pakeičia iškviečiamo modelio būseną. Interpretuojamos pelės arba klaviatūros įvestys iš varotojo, pagal jas yra atliekami modelyje arba / ir vaizde atitinkami pakeitimai.

Kontrolerio pagalba vartotojas sąveikauja su aplikacija. Kontroleris priima vartotojo įvestį ir duoda nurodymus modeliui ir vaizdui įvykdyti veiksmus pagal pateiktą įvestį.

2.5.4 Komponentų tarpusavio sąryšiai

Modelis, vaizdas ir kontroleris yra artimai vienas su kitu susiję ir nuolat tarpusavyje kontaktuojantys. 5 pav. pateikta MVC paradigmos ryšių iliustracija [7]. Gana sunku yra pasiekti griežtą modelio ir vaizdo atskyrimą.



5 pav. MVC modelis

Faktas, kad reikia atskirti srities logiką nuo prezentacijos yra beveik neginčytina objektiškai orientuoto projektavimo aksioma, bet lyg šiol daugelis sėkmingų projektų šiuos metodikos netaiko.

Raginimas nenaudoti MVC paradigmos grindžiamas sudėtingumu ir sunkumu naudotis. Praktikoje griežtas trijų MVC pagrindinių komponentų atskyrimas yra sunkiai pasiekiamas arba neįmanomas. Dažniausiai rekomenduojama aplikaciją sudalinti į komponentus atsakingus už įėjimą, vykdymą ir išėjimą.

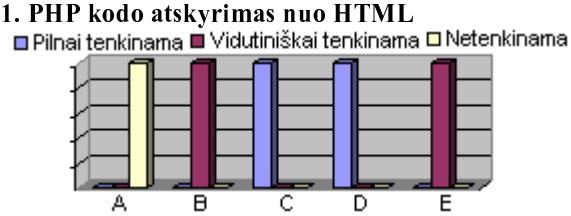
3. Šablonų sistemų vertinimas ir išvados

Norint įvertinti šablonų sistemas pagal mano įvestą taksonomiją, pirmiausiai reikia nusistatyti vertinimo metrikas 1 lentelė. Panaudodamas šias metrikas pateikiu šablonų sistemų vertinimą.

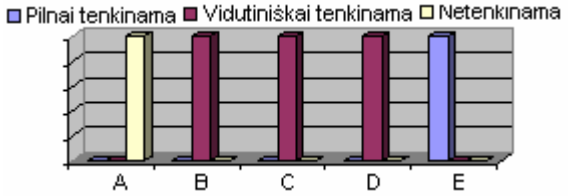
Žymėjimai:

A - Standartinėmis priemonėmis paremtas šablonų įgyvendinimas
B - Paprastos šablonų sistemos
C - Šablonų sistemos palaikančios kompiliavimą ir papildomus modulius
D - XSLT transformacijomis paremta šablonų sistema
E - Modelis – Vaizdas – Kontroleris struktūrinės šablonų sistemos

1. PHP kodo atskyrimas nuo HTML

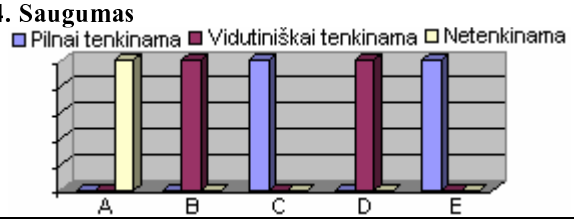


2. Srities logikos atskyrimas nuo atvaizdavimo

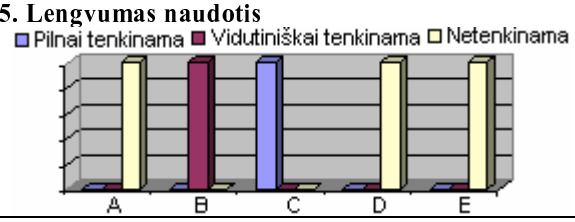


3. Šablono papildomų šablono savybių palaikymas

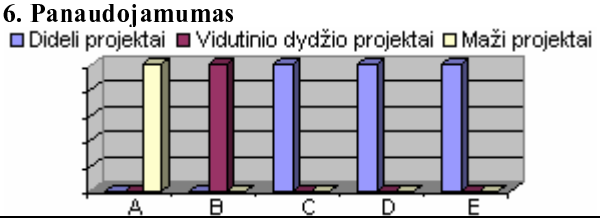
4. Saugumas



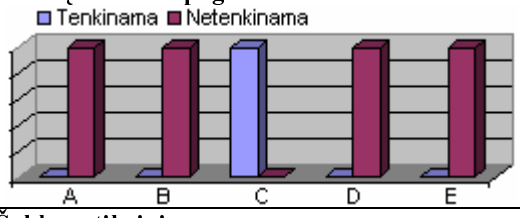
5. Lengvumas naudotis



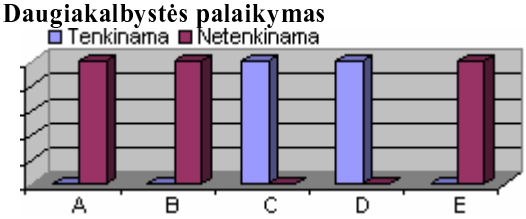
6. Panaudojamumas



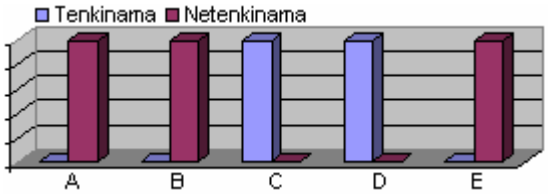
7. Klaidų šalinimo pagalba



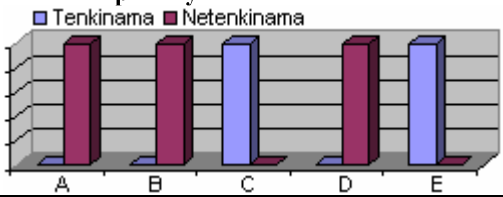
8. Daugiakalbystės palaikymas



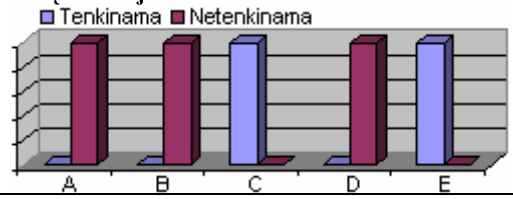
9. Šablono tikrinimas



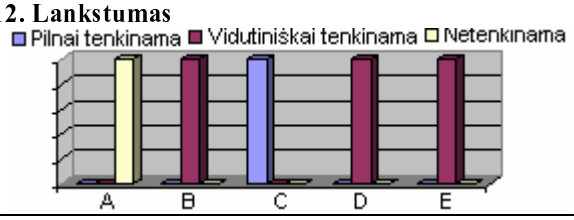
10. Kešavimo palaikymas



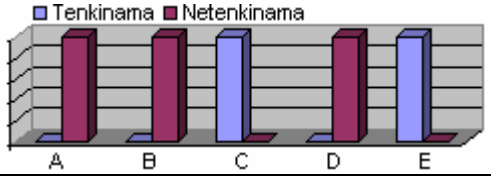
11. Savęs stebėjimas



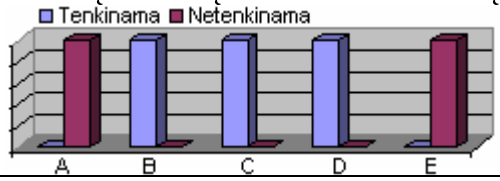
12. Lankstumas



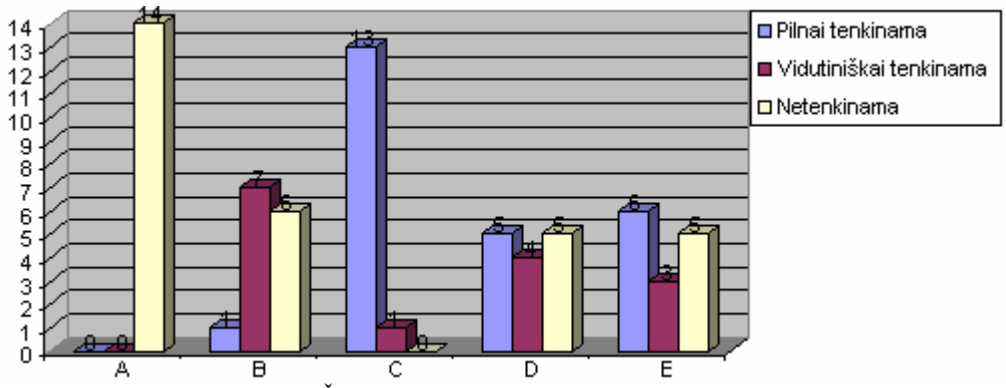
13. Kodo dubliavimas



14. Duomenų struktūrų eliminavimas iš šablonų



Apibendrintas šablonų sistemų vertinimas pagal mano išskirtas metrikas pateiktas 6 pav.



6 pav. Šablonų sistemų apibendrintas vertinimas

Literatūros sąrašas

1. LAURENCE, M. Rapid Template-Driven Web Development. *The Tenth International World Wide Web Conference.* Hong Kong, 2001.
2. MULYE, R. A Semantic Template Based Designer for Web Processes. *IEEE International Conference on Web Services*. Florida, USA, 2005.
3. GANGEMI, G. Ontology Design Patterns for Semantic Web Content. *The SemanticWeb - ISWC 2005, 4th International SemanticWeb Conference*. Galway, Ireland, 2005, p. 262-276 p.
4. ZHIMA, G.; MIN, L.; XIAOLING, W. Scalable XSLT Evaluation. *Advanced Web Technologies and Applications: 6th Asia-Pacific Web Conference.* Hangzhou, China, 2004, p. 190 – 200.
5. REENSKAUG, T. The Model-View-Controller (MVC) Its Past and Present Trygve Reenskaug. *JAOO Conference*. Aarhus, Denmark, 2003.
6. BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; STAL, M. Pattern -Oriented Software Architecture. New York, 2000.
7. GRESH, J. E. The Collection Switch Design Pattern. *Rensselaer Computer Science Conference*. Oslo, 2004.

WEB TEMPLATE SYSTEMS TAXONOMY

Complex Web pages development forces to use new technologies that help in development phases. It is possible to separate business logic from presentation using template engines. It's not easy to choose a proper template engine system. I classified template systems and showed each classification group analysis and results.

2 priedas. Panaudojimo atvejų sąrašas

1. PANAUDOJIMO ATVEJIS: Praktikos darbų registravimas	
Vartotojai / Aktoriai:	Įmonės atstovas, Duomenų saugyklos
Aprašas:	Procesas kurio metu yra pasiūlomi praktikos darbai.
Prieš sąlyga:	1. Aktorius turi būti užregistruotas sistemoje; 2. Aktorius pateikiąs teisingą slaptažodį turi būti prisijungęs į sistemą.
Sužadinimo sąlyga:	1. Aktorius turi pasirinkti praktikos darbų registravimo meniu skiltį.
Po-sąlyga:	1. Praktikos darbas įtraukiamas į praktikos darbų sąrašą;

2. PANAUDOJIMO ATVEJIS: Praktikos darbų patvirtinimas	
Vartotojai / Aktoriai:	Dėstytojas, Duomenų saugyklos
Aprašas:	Tikslas – studentui priskirti tinkamą praktikos darbą.
Prieš sąlyga:	1. Aktorius turi būti užregistruotas sistemoje; 2. Aktorius pateikiąs teisingą slaptažodį turi būti prisijungęs į sistemą.
Sužadinimo sąlyga:	1. Aktorius turi pasirinkti praktikos darbų patvirtinimo meniu skiltį.
Po-sąlyga:	1. Studentas gali atlikti praktiką pasirinktame darbe.

3. PANAUDOJIMO ATVEJIS: Praktikos darbų modifikavimas	
Vartotojai / Aktoriai:	Dėstytojas, Įmonės atstovas, Duomenų saugyklos
Aprašas:	Informacijos susijusios su pasiūlytu praktikos darbu keitimas. Įmonės atstovas gali keisti tik savo pasiūlytų praktikos darbų informaciją.
Prieš sąlyga:	1. Aktorius turi būti užregistruotas sistemoje; 2. Aktorius pateikiąs teisingą slaptažodį turi būti prisijungęs į sistemą.
Sužadinimo sąlyga:	1. Aktorius turi pasirinkti praktikos darbų modifikavimo meniu skiltį.
Po-sąlyga:	1. Sena informacija apie praktikos darbą pakeičiama nauja.

4. PANAUDOJIMO ATVEJIS: Praktikos darbų pasirinkimas	
Vartotojai / Aktoriai:	Studentas, Duomenų saugyklos
Aprašas:	Studentas išsirenka sau patinkantį praktikos darbą.
Prieš sąlyga:	1. Aktorius turi būti užregistruotas sistemoje; 2. Aktorius pateikiąs teisingą slaptažodį turi būti prisijungęs į sistemą.
Sužadinimo sąlyga:	1. Aktorius turi pasirinkti praktikos darbų pasirinkimo meniu skiltį.
Po-sąlyga:	1. Pasirinktas praktikos darbas perduodamas dėstytojo patvirtinimui.

5. PANAUDOJIMO ATVEJIS: Praktikos darbų peržiūra	
Vartotojai / Aktoriai:	Dėstytojas, Įmonės atstovas, Studentas, Duomenų saugyklos
Aprašas:	Studentas ir dėstytojas gali stebėti visus sistemoje esančius praktikos darbus, jų aprašymus ir komentarus. Įmonės atstovas mato tik savo pasiūlytus darbus.
Prieš sąlyga:	1. Aktorius turi būti užregistruotas sistemoje; 2. Aktorius pateikiąs teisingą slaptažodį turi būti prisijungęs į sistemą.
Sužadinimo sąlyga:	1. Aktorius turi pasirinkti praktikos darbų peržiūros meniu skiltį.
Po-sąlyga:	1. Pateikiamas praktikos darbų sąrašas.

6. PANAUDOJIMO ATVEJIS: Vartotojų valdymas	
Vartotojai / Aktoriai:	Dėstytojas, Studentas, Įmonės atstovas, Duomenų saugyklos, Sistemos administratorius
Aprašas:	Vartotojo informacijos keitimas. Galimybė pasikeisti savo kontaktinę informaciją
Prieš sąlyga:	1. Aktorius turi būti užregistruotas sistemoje; 2. Aktorius pateikiąs teisingą slaptažodį turi būti prisijungęs į sistemą.
Sužadinimo sąlyga:	1. Aktorius turi pasirinkti vartotojų informacijos keitimo meniu skiltį; 2. Slaptažodžio galiojimo pabaiga.
Po-sąlyga:	1. Sena informacija pakeičiama nauja.

7. PANAUDOJIMO ATVEJIS: Vartotojų peržiūra	
Vartotojai / Aktoriai:	Dėstytojas, Įmonės atstovas, Studentas, Duomenų saugyklos, Sistemos administratorius
Aprašas:	Vartotojo informacijos peržiūra.
Prieš sąlyga:	1. Aktorius turi būti užregistruotas sistemoje; 2. Aktorius pateikiąs teisingą slaptažodį turi būti prisijungęs į sistemą.
Sužadinimo sąlyga:	1. Aktorius turi pasirinkti vartotojų informacijos peržiūros meniu skiltį.
Po-sąlyga:	1. Užklausos informacija apie vartotojus pateikiama aktoriui.

8. PANAUDOJIMO ATVEJIS: Bendravimas tarp vartotojų	
Vartotojai / Aktoriai:	Dėstytojas, Studentas, Įmonės atstovas, Duomenų saugyklos
Aprašas:	Tikslas – sukurti vidinę komunikaciją tarp vartotojų.
Prieš sąlyga:	1. Aktorius turi būti užregistruotas sistemoje; 2. Aktorius pateikiąs teisingą slaptažodį turi būti prisijungęs į sistemą.
Sužadinimo sąlyga:	1. Aktorius turi pasirinkti žinučių meniu skiltį. 2. Aktorius pasirenka praktikos darbų peržiūros meniu skiltį ir turi galimybę perskaityti komentarus apie siūlomą praktikos darbą bei parašyti komentarą.

Po-sąlyga:	1. Aktorius nusiunčia arba perskaito žinutes.

9. PANAUDOJIMO ATVEJIS: Sistemos administravimas	
Vartotojai / Aktoriai:	Sistemos administratorius, Duomenų saugyklos
Aprašas:	Sistemos administratorius turi užtikrinti sistemos apvalymą, saugumą, stabilumą, Studentų, dėstytojų, įmonės atstovų, administratorių informacijos modifikavimas, universitetų duomenų valdymas.
Prieš sąlyga:	1. Aktorius turi būti užregistruotas sistemoje; 2. Aktorius pateikiąs teisingą slaptažodį turi būti prisijungęs į sistemą.
Sužadinimo sąlyga:	1. Aktorius turi pasirinkti sistemos valdymo meniu skiltį.
Po-sąlyga:	1. Sistemoje atliekamas administravimo veiksmas.

3 priedas. Duomenų bazės lentelių aprašymai

Lentelė students skirta studentų informacijai saugoti. Lentelės aprašymas pateiktas 16 lentelėje.

16 lentelė. students

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Studento identifikacinis numeris
student_no	Varchar(20)	Ne	Studento pažymėjimo numeris
name	Varchar(100)	Ne	Studento vardas
surname	Varchar(100)	Ne	Studento pavardė
username	Varchar(100)	Ne	Studento vartotojo vardas
password	Varchar(100)	Ne	Studento Slaptažodis
email	Varchar(100)	Ne	Studento elektroninis paštas
phone	Varchar(15)	Taip	Studento telefono numeris
flag	Smallint	Taip	Papildoma studento nuostata
speciality_id (Foreign key)	Integer	Ne	Studento specialybės identifikacinis numeris
block	Boolean	Ne	Studento prieigos prie darbų portalo blokavimo parametras

Lentelė teachers skirta dėstytojų informacijai saugoti. Lentelės aprašymas pateiktas 17 lentelėje.

17 lentelė. teachers

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Dėstytojo identifikacinis numeris
name	Varchar(100)	Ne	Dėstytojo vardas
surname	Varchar(100)	Ne	Dėstytojo pavardė
department_id (Foreign	Integer	Ne	Katedros identifikacinis

key)			numeris
username	Varchar(100)	Ne	Dėstytojo vartotojo vardas
password	Varchar(100)	Ne	Dėstytojo slaptažodis
email	Varchar(100)	Ne	Elektroninis paštas
phone	Varchar(15)	Ne	Telefonas

Lentelė companies skirta įmonių informacijai saugoti. Lentelės aprašymas pateiktas 18 lentelėje.

18 lentelė. companies

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Įmonės identifikacinis numeris
name	Varchar(100)	Ne	Įmonės pavadinimas
contact_person_name	Varchar(100)	Ne	Įmonės kontaktinio asmens vardas
contact_person_surname	Varchar(100)	Ne	Įmonės kontaktinio asmens pavardė
activity_desc	Text	Ne	Trumpas įmonės veiklos aprašymas
username	Varchar(100)	Ne	Įmonės vartotojo vardas
password	Varchar(100)	Ne	Įmonės slaptažodis
email	Varchar(100)	Ne	Įmonės elektroninis paštas
phone	Varchar(15)	Ne	Įmonės telefonas
fax	Varchar(15)	Ne	Įmonės faksas
flag	Smallint	Taip	Papildoma įmonės nuotata
contact_person_phone	Varchar(15)	Ne	Įmonės kontaktinio asmens telefono numeris
contact_person_email	Varchar(100)	Ne	Įmonės kontaktinio asmens elektroninio pašto adresas
code	Char(9)	Ne	Įmonės kodas
address	Varchar(200)	Ne	Įmonės adresas
registre_date	Date	Ne	Įmonės registracijos data
chief_role	Varchar(100)	Ne	Įmonės vadovo pareigos
chief_name	Varchar(100)	Ne	Įmonės vadovo vardas
chief_surname	Varchar(100)	Ne	Įmonės vadovo pavardė
block	Boolean	Ne	Įmonės prieigos prie darbų portalo blokavimo parametras

Lentelė admins skirta administratorių informacijai saugoti. Lentelės aprašymas pateiktas 19 lentelėje.

19 lentelė. admins

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Administratoriaus identifikacinis numeris
name	Varchar(100)	Ne	Administratoriaus vardas
surname	Varchar(100)	Ne	Administratoriaus pavardė

username	Varchar(100)	Ne	Administratoriaus vartotojo vardas
password	Varchar(100)	Ne	Administratoriaus slaptažodis
email	Varchar(100)	Ne	Administratoriaus elektroninio pašto adresas
phone	Varchar(15)	Ne	Administratoriaus telefono numeris

Lentelė categories skirta įmonės veiklos kategorijoms saugoti. Lentelės aprašymas pateiktas 20 lentelėje.

20 lentelė. categories

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Įmonės veiklos kategorijos identifikacinis numeris
name	Varchar(200)	Ne	Įmonės veiklos pavadinimas

Lentelė subcategories skirta įmonės veiklos sub-kategorijoms saugoti. Lentelės aprašymas pateiktas 21 lentelėje.

21 lentelė. subcategories

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Įmonės veiklos sub-kategorijos identifikacinis numeris
name	Varchar(200)	Ne	Įmonės veiklos pavadinimas
category_id (Foreign key)	Integer	Ne	Įmonės veiklos kategorijos identifikacinis numeris

Lentelė company_subcategories skirta įmonės pasirinktoms veiklos sub-kategorijoms saugoti. Lentelės aprašymas pateiktas 22 lentelėje.

22 lentelė. company_subcategories

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Įmonės pasirinktos veiklos sub-kategorijos identifikacinis numeris
subcategory_id (Foreign key)	Integer	Ne	Įmonės veiklos sub-kategorijos identifikacinis numeris
company_id (Foreign key)	Integer	Ne	Įmonės identifikacinis numeris

Lentelė universities skirta universitetų duomenims saugoti. Lentelės aprašymas pateiktas 23 lentelėje.

23 lentelė. universities

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Universiteto identifikacinis numeris
name	Varchar(100)	Ne	Universiteto pavadinimas

city	Varchar(100)	Ne	Miestas kuriame yra universitetas
street	Varchar(100)	Ne	Gatvė kurioje yra universitetas
street_no	Varchar(40)	Ne	Pastato numeris kuriame yra universitetas
code	Varchar(20)	Ne	Gatvės kodo kurioje yra universitetas
company_code	Char(9)	Ne	Universiteto įmonės kodas
phone	Varchar(15)	Ne	Universiteto telefono numeris
fax	Varchar(15)	Ne	Universiteto fakso numeris
email	Varchar(100)	Ne	Universiteto elektroninio pašto adresas
www	Varchar(200)	Ne	Universiteto interneto svetainės adresas

Lentelė faculties skirta fakultetų duomenims saugoti. Lentelės aprašymas pateiktas 24 lentelėje.

24 lentelė. faculties

Stulpelio pavadinimas	**Tipas**	**Ar leidžiama NULL**	**Aprašymas**
ID (Primary key)	Integer	Ne	Fakulteto identifikacinis numeris
name	Varchar(100)	Ne	Fakulteto pavadinimas
city	Varchar(100)	Ne	Miestas kuriame yra fakultetas
street	Varchar(100)	Ne	Gatvė kurioje yra fakultetas
street_no	Varchar(40)	Ne	Namo numeris kuriame yra fakultetas
code	Varchar(20)	Ne	Gatvės kodo numeris kuriame yra fakultetas
phone	Varchar(15)	Ne	Fakulteto telefono numeris
fax	Varchar(15)	Ne	Fakulteto fakso numeris
email	Varchar(100)	Ne	Fakulteto elektroninio pašto adresas
www	Varchar(200)	Ne	Fakulteto interneto svetainės adresas
university_id (Foreign key)	Integer	Ne	Universiteto kuriame yra fakultetas identifikacinis numeris

Lentelė department skirta katedrų duomenims saugoti. Lentelės aprašymas pateiktas 25 lentelėje.

25 lentelė. department

Stulpelio pavadinimas	**Tipas**	**Ar leidžiama NULL**	**Aprašymas**
ID (Primary key)	Integer	Ne	Katedros identifikacinis numeris
name	Varchar(100)	Ne	Katedros pavadinimas

city	Varchar(100)	Ne	Miestas kuriame yra katedra
street	Varchar(100)	Ne	Gatvė kurioje yra katedra
street_no	Varchar(40)	Ne	Namo numeris kuriame yra katedra
code	Varchar(20)	Ne	Gatvės kodo numeris kuriame yra katedra
phone	Varchar(15)	Ne	Katedros telefono numeris
fax	Varchar(15)	Ne	Katedros fakso numeris
email	Varchar(100)	Ne	Katedros elektroninio pašto adresas
www	Varchar(200)	Ne	Katedros interneto svetainės adresas
chief_name	Varchar(100)	Ne	Katedros vadovo vardas
chief_surname	Varchar(100)	Ne	Katedros vadovo pavardė
chief_email	Varchar(100)	Ne	Katedros vadovo elektroninio pašto adresas
chief_phone	Varchar(15)	Ne	Katedros vadovo telefono numeris
faculty_id (Foreign key)	Integer	Ne	Fakulteto kuriame yra katedra identifikacinis numeris

Lentelė teachers_modules dėstytojams priskirtų praktikos modulių informacijai saugoti. Lentelės aprašymas pateiktas 26 lentelėje.

26 lentelė. teachers_modules

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Dėstytojui priskirto praktikos modulio identifikacinis numeris
teacher_id (Foreign key)	Integer	Ne	Dėstytojo identifikacinis numeris
practice_module_id (Foreign key)	Integer	Ne	Praktikos modulio identifikacinis numeris
closed	Boolean	Ne	Leidimas prisijungti į šį praktikos modulį

Lentelė tasks skirta dėstytojų sukurtų užduočių studentams informacijos saugojimui. Lentelės aprašymas pateiktas 27 lentelėje.

27 lentelė. tasks

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Dėstytojo sukurtos užduoties identifikacinis numeris
teacher_module_id (Foreign key)	Integer	Ne	Dėstytojui priskirto praktikos modulio identifikacinis numeris
task_name	Varchar(200)	Ne	Užduoties antraštė
task_description	Text	Ne	Užduoties aprašymas

Lentelė students_reports skirta studentų ataskaitų saugojimui. Lentelės aprašymas pateiktas 28 lentelėje.

28 lentelė. student_reports

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Studento užduoties identifikacinis numeris
task_id (Foreign key)	Integer	Ne	Dėstytojo užduoties identifikacinis numeris
title	Varchar(200)	Ne	Studento ataskaitos antraštė
description	Text	Ne	Studento ataskaita
student_id	Integer	Ne	Studento identifikacinis numeris
grade	Smallint	Taip	Ataskaitos įvertinimas

Lentelė students_reports_comments skirta studentų ataskaitų komentarų saugojimui. Lentelės aprašymas pateiktas 29 lentelėje.

29 lentelė. student_reports_comments

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Studento ataskaitos komentaro identifikacinis numeris
student_report_id (Foreign key)	Integer	Ne	Studento ataskaitos identifikacinis numeris
comment	Text	Ne	Komentaras
time	Timestamp	Ne	Komentaro įrašymo laikas
commenter_id	Integer	Ne	Komentuoto identifikacinis numeris (Dėstytojas, studentas)
commenter_table_id	Integer	Ne	Komentuotojo lentelė (ar studentas ar dėstytojas)

Lentelė messages skirta žinutėms saugoti. Lentelės aprašymas pateiktas 30 lentelėje.

30 lentelė. messages

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Žinutės identifikacinis numeris
from_id (Foreign key)	Integer	Ne	Žinutės rašytojo identifikacinis numeris
from_person_table	Varchar(20)	Ne	Žinutės rašytojo lentelės pavadinimas
to_id (Foreign key)	Integer	Ne	Žinutės gavėjo identifikacinis numeris
to_person_table	Varchar(20)	Ne	Žinutės gavėjo lentelės pavadinimas
message	Text	Ne	Žinutės tekstas
send_time	Date	Ne	Žinutės išsiuntimo laikas

flag	Smallint	Ne	Žinutės papildomas atributas
teacher_id (Foreign key)	Integer	Taip	Dėstytojo identifikacinis numeris
company_id (Foreign key)	Integer	Taip	Įmonės identifikacinis nuemris
admin_id (Foreign key)	Integer	Taip	Administratoriaus identifikacinis numeris
student_id (Foreign key)	Integer	Taip	Studento identifikacinis numeris

Lentelė practice_jobs skirta siūlomiems praktikos darbams saugoti. Lentelės aprašymas pateiktas 31 lentelėje.

31 lentelė. practice_jobs

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Siūlomo praktikos darbo identifikacinis numeris
title	Varchar(100)	Ne	Praktikos darbo antraštė
description	Text	Ne	Praktikos darbo aprašymas
company_id (Foreign key)	Integer	Ne	Įmonės identifikacinis numeris
subcategory_id (Foreign key)	Integer	Ne	Sub-kategorijos identifikacinis numeris

Lentelė practice_jobs_comments skirta siūlomų praktikos darbų komentarams saugoti. Lentelės aprašymas pateiktas 32 lentelėje.

32 lentelė .practice_jobs_comments

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Praktikos darbo komentaro identifikacinis numeris
commenter_id (Foreign key)	Integer	Ne	Komentuoto identifikacinis numeris (Dėstytojas, studentas, įmonė)
commenter_table_id	Integer	Ne	Komentuotojo lentelė (studentas, įmonė ar dėstytojas)
comment	Text	Ne	Praktikos darbo komentaras
time	Date	Ne	Praktikos darbo komentaro įrašymo laikas
practice_job_id (Foreign key)	Integer	Ne	Praktikos darbo identifikacinis numeris

Lentelė choised_jobs skirta pasirinktų darbų informacijai saugoti. Lentelės aprašymas pateiktas 33 lentelėje.

33 lentelė. choised_jobs

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Pasirinkto praktikos darbo identifikacinis numeris

student_id (Foreign key)	Integer	Ne	Studento identifikacinis numeris
practice_job_id (Foreign key)	Integer	Ne	Praktikos darbo identifikacinis numeris
teacher_module_id (Foreign key)	Integer	Ne	Dėstytojo administruojamo praktikos modulio identifikacinis numeris

Lentelė choised_job_comments skirta pasirinktų darbų komentarams saugoti. Lentelės aprašymas pateiktas 34 lentelėje.

34 lentelė. choised_job_comments

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Pasirinkto praktikos darbo komentaro identifikacinis numeris
choised_job_id (Foreign key)	Integer	Ne	Pasirinkto praktikos darbo identifikacinis numeris
comment	Text	Ne	Komentaras
time	Date	Ne	Pasirinkto praktikos darbo komentaro įrašymo laikas
commenter_id (Foreign key)	Integer	Ne	Komentuoto identifikacinis numeris (Dėstytojas, studentas, įmonė)
commenter_table_id	Integer	Ne	Komentuotojo lentelė (studentas, įmonė ar dėstytojas)

Lentelė speciality skirta specialybių aprašymui saugoti. Lentelės aprašymas pateiktas 35 lentelėje.

35 lentelė. speciality

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Specialybės identifikacinis numeris
name	Varchar(100)	Ne	Specialybės pavadinimas
description	Text	Ne	Specialybės aprašymas
department_id (Foreign key)	Integer	Ne	Katedros identifikacinis numeris

Lentelė practice_modules skirta praktikos modulių aprašymui saugoti. Lentelės aprašymas pateiktas 36 lentelėje.

36 lentelė. practice_modules

Stulpelio pavadinimas	Tipas	Ar leidžiama NULL	Aprašymas
ID (Primary key)	Integer	Ne	Praktikos modulio identifikacinis numeris

code	Varchar(100)	Ne	Praktikos modulio kodas
study_program	Varchar(100)	Ne	Praktikos modulio studijų programa
name	Varchar(100)	Ne	Praktikos modulio pavadinimas
requirement	Varchar(100)	Ne	Praktikos modulio reikalavimai
description	Text	Ne	Praktikos modulio aprašymas
goal	Text	Ne	Praktikos modulio tikslas
speciality_id (Foreign key)	Integer	Ne	Specialybės identifikacinis numeris