LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

PHP IETVARI TĪMEKĻA VIETŅU IZSTRĀDEI

BAKALAURA DARBS

Autors: **Edmunds Beinarovičs**
Studenta apliecības Nr.: eb09078
Darba vadītājs: profesors Dr. Dat. Kārlis Čerāns

RĪGA 2013

SATURA RĀDĪTĀJS

ANOTĀCIJA

Šinī darbā tiek īsi apskatīti ietvari tīmekļa izstrādei, PHP programmēšanas valodā. Tiek izvēlēti un pētīti divi ietvari smalkākai analīzei, Symfony2 un Zend Framework 2.

Pētījumam tiek izstrādātas divas tīmekļa vietnes, pa vienai katrā no izvēlētiem ietvariem. To izstrādes gaitā, tiek apskatīti ietvaru rīki, visbiešāk sastopamiem uzdevumiem, tādiem kā saskarsme ar datu bāzi, veidlapu apstrāde un skatu pārvaldība.

Kā svarīgs vērtēšanas kritērijs ir ietvara saprotamība un paātrinoši izstrādes gaitu piedāvātie palīgrīki. Balstoties uz šiem kritērijiem tiek salīdzināti ietvari un analizējot sasniegti secinājumi.

ANOTATION

This paper contains a short view of a few frameworks based on PHP programming language. In conclusion, Symfony2 and Zend Framework 2 are selected for future analyze.

To study the selected frameworks, two websites where developed, one for each framework. It contains the review of the most useful framework parts like database management, form and view management.

The most important criteria for conclusions are the frameworks understandability and utilities for development flows acceleration. Based on these criteria, the frameworks are compared and conclusions are made.

APZĪMĒJUMU SARAKSTS

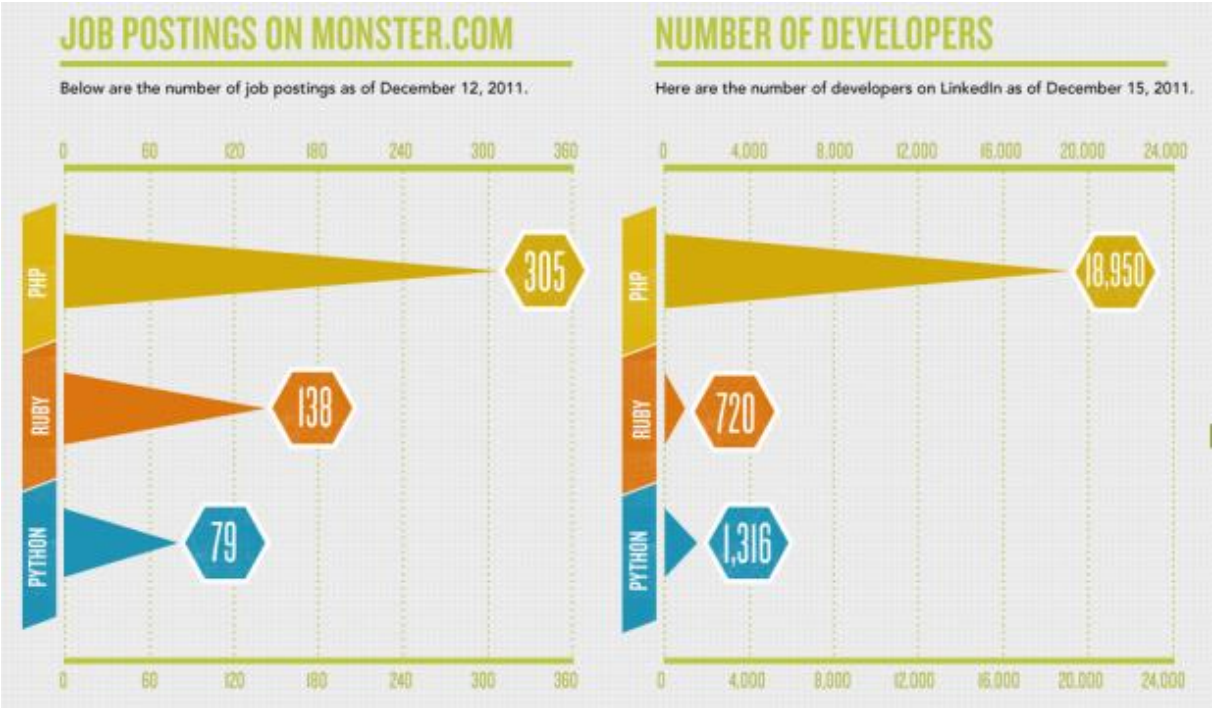
ZF2	Zend Framework 2 ietvars.
Injekcija	Automātiskā objektu vai vērtību padošana.
Komponente	Atsevišķa, visbeižāk neatkarīga pakete, kuru var pievienot projektam.
ORM	Object Based Mapping – izmantojas, lai savienot klases ar datu bāzi.
DQL	Doctrine Query Language – Objektorientēta pieprasījumu valoda.
MVC	Model, View, Controller – Standarts kā dalīt tīmekļa izstrādi trijās daļās.
URI	Uniform Resource Identifier – Tīmekļa adreses segments, identificējošs to. Piemēram „http://zend.beinarovic.lv/list" adresei ir viens URI segments „list".
Serviss	Serviss ir kāda atsevišķa klase, vai klašu kopums kurš veic kādus darbus, un kuru izsauc pamatklase.
Konsole	Programmatūra, kas izpilda komandrindā ievadītās komandas, ar teksta interfeisu.
Konsoles komanda	Vienas rindas uzdevums ievadīts komandrindā.
Asset	Ārēji pieejama datne. Piemērs – CSS datnes.
Composer	Atkarību pārvaldību rīks, kurš instalē komponentes.
Kontrolieris	Funkcija kura saņem HTTP pieprasījumu un izveido HTTP atbildi kuru atgriež. Kā arī kontrolieris var atbildes vietā pāradresēt uz citu maršrutu.
SOLID	Pieci standarti, kas ir atbildīgi par koda pareizrakstību.
Anotācijas	PHP programmēšanas vidē, komentārā, pierakstīti uzstādījumi.

IEVADS

Tīmekļa tehnoloģijas attīstās ļoti strauji, gandrīz visa pasaule izmanot tīmekli lai piekļūtu kādai informācijai vai lai paveiktu kautko. Ir dažādi tīmekļa lietotņu veidošanas ietvari, kuri katrs piedāvā mazliet atšķirīgas iespējas tīmekļa lietotnes izveidei, un katrs akcentē galvenokārt savas pozitīvās puses. Šī darba mērķis ir pārliecināties par nelielas praktiskas sistēmas izstrādes iespējamību un vieglumu konkrētos tīmekļa lietotņu izstrādes ietvaros, kā arī veikt izmantoto ietvaru salīdzināšanu.

Kā vietnes sfēra izvēlēta ir „Uzdevumu vadības sistēma", kurā var veidot lietotājus, strādāt ar uzdevumiem, kā arī grupēt tos.

Līderi starp programmēšanas valodām tīmeklim ir PHP, Python, Ruby un Node JS. Apskatot šīs valodas, tika izlemts pielietot PHP programmēšanas valodu. Jo PHP ir pirmā programmēšanas valoda tēmēta uz tīmekļa vietņu izstrādi. Kā arī tai pieder bagāta dokumentācija tīmeklī, to joprojām uzlabo, un sagaida jaunas versijas. No pāša sākuma līdz pat šīm dienām līderis starp darbu vakancēm tīmekļa izstrādei arī ir PHP valodas programmētājs.



Ilustrācija 1 Darbu vakanču un izstrādātāju skaita attēlojošais grafiks, balstoties uz rakstu "Code Wars: Ruby vs Python vs PHP".

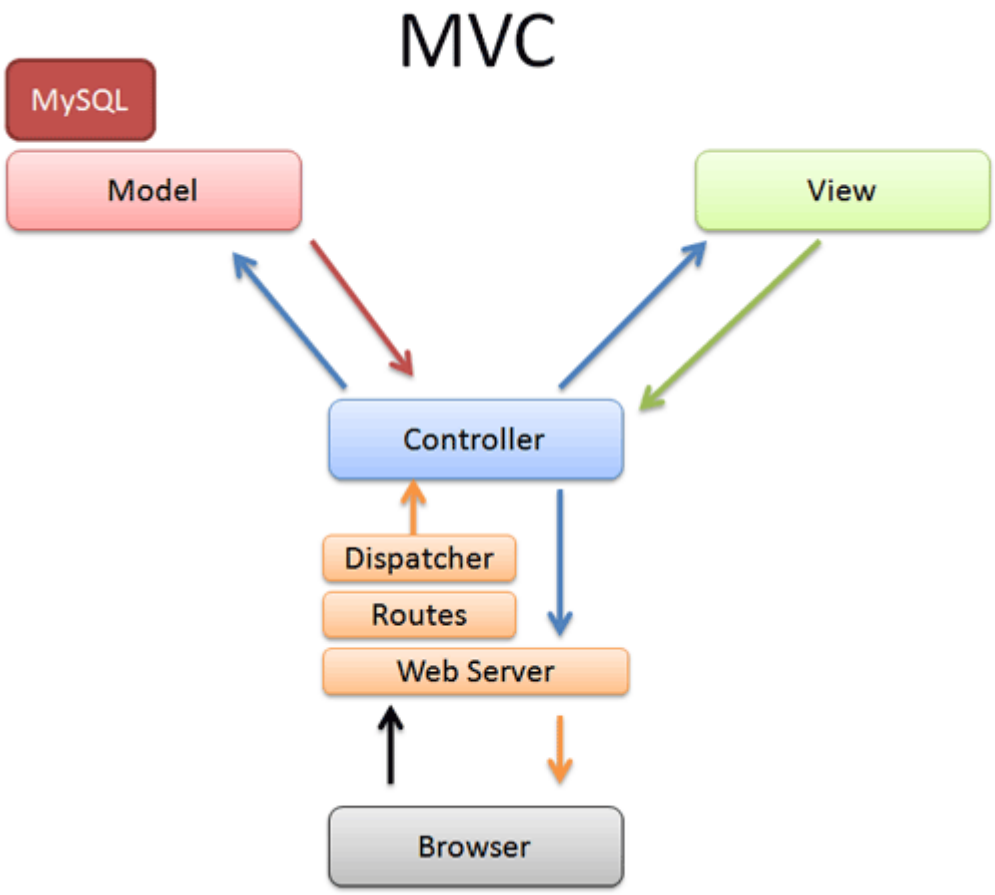
Tajā vidē kurā visvairāk tiek darīts, jābūt visvairāk resursu un jau gatavu materiālu. Tas paātrina izstrādi, jo labāk ir dabūt jau gatavu risinājumu nekā izveidot kautko kas jau ir bijis paveikts.

1. IETVARU IZVĒLE

Tika apskatīvi vairāki ietvari PHP programmēšanas valodas vidē. Izvēlētie līderi ir – Symfony2, Zend Framework 2, Yii Framework, CodeIgniter un Cake PHP. Izvēle tika veikta balstoties uz atsauksmēm. Visi izvēlētie ietvari izņemot objektorientēto programmēšanu un izņemot CodeIgniter, kurā tā parādās, bet reti.

Symfony2 ietvars	Izmanto konsoli, kas nozīmē ka kādi darbi tiks veikti automātiski. Komplektā iet ar „Doctrine2" kas ir populārs un labi dokumentēts ORM rīks. Nav vienkārši to instalēt un uzstādīt. Ļoti labi der lieliem projektiem. Tam tiek piedāvāta liela komponenšu bibleotēka.
Zend Framework 2 ietvars	Šo ietvaru izstrādāja PHP valodas izstrādātāji. Tas ir ļoti elastīgs, to var pielāgot jebkādam projektam. Tam ir iebūvēts ORM rīks, bet pēc atsauksmēm tas nav īpaši ērts. Priekš tā ir vairākas komponentes.
Yii Framework ietvars	Diezgan vienkāršs apmācībai, piemīt koda ģenerēšana. Izmanto „Yii Active Record" kā savu ORM rīku. Tas ir paredzēts gan uz vienkāršiem projektiem gan uz lieliem. Imanto komponentes.
CodeIgniter ietvars	Ļoti vienkāršs un ar labu dokumentāciju. Tas minimāli izmanto objektorientēto PHP. Ļoti labs nelielu projektu izstrādei.
Cake PHP ietvars	Tam ir iebūvēts ORM rīks, pēc atsauksmēm tas nav praktisks. To var ļoti vienkārši uzstādīt un sākt ar to strādāt.

Visi šie ietvari, izņemot Symfony2, balstās uz MVC principa, kas ir princips kurš atdala skatus, modeļus un kontrolierus.



Ilustrācija 1.1 Shēmā tiek parādīta datu plūsma MVC vidē.

1.1. Izvēlētie ietvari pētījumam

Pirmkārt tika izvēlēts Symfony2 ietvars, jo par to varēja sameklēt ļoti daudz informācijas. Atsauksmes bija ļoti labas. Un tajā ietilpst viss nepieciešams vienkāršai tīmekļa vietnes izstrādei. Symfony2 pa tiešo strādā ar Doctrine2, par kuru visi raksta tikai labas atsauksmes.

Balstoties uz atsauksmēm, gan Zend Framework 2, gan Yii, ir vērts pētīt. Yii pēc apraksta šķita ļoti vienkāršs un saturīgs, tomēr tika izvēlēts ZF2. Balstoties uz to ka to izstrādāja tā pati komanda kura izstrādā pašu PHP programmēšanas valodu.

Abi ietvari ir lielākoties tēmēti uz lielākiem projektiem, bet abi sola ārkārtīgu ātrdarbību. Izstrādājot tos mēģināja pieturēties pie SOLID standartiem, kas liecina par to, ka viņu sākotnējam kodam ir jābūt viegli saprotamam.

1.1.1. Symfony2 ietvars

Symfony2 tika izvēlēts jo pēc atziņām tas ir ļoti spēcīgs un pieredzējošiem lietotājiem ļoti ērts. Kā arī tam ir ļoti daudz komponentu, kas nozīmē, ka varēs atrast gatavus risinājumus kādām situācijām.

Tam ir stingri noteikta struktūra kas liek tā lietotājiem sekot standartiem. Standartu izmantošana atbalsta projekta saprotamību. Tas ir elastīgs tādēļ, ka tam ir iespējas pārrakstīt servisus nevis veidot sevis izdomāto struktūru. Rakstot pēc stadartiem citi lietotāji varēs tikpat labi saprast kas notiek kodā.

„Sensio Labs" dod 3 gadu garantiju izlaizšanas brīdī, uzturēt un labot izlaisto versiju.

Symfony2 ir iebūvētā izstrādes vide, kura parāda tekošo stāvokli, kā arī visu pieprasījumu vēsturi.

Izstrādājot tīmekļa vietnes uz Symfony2, praktiski nav iespējams dot iespēju lietotājiem uzlauzt datu bāzi pielietojot SQL injeksijas, jo automātiski notiek maršrutēšanas filtrēšana un Doctrine2 filtrē visus pieprasījuma datus.

1.1.2. Zend Framework 2 ietvars

Galvenais izvēles kritērijs kādēļ ZF2 tika izvēlēts, bija tāds, ka to izveidoja tā pati komanda, kura izveidoja pašu programmēšanas valodu PHP. Kā arī balstoties uz to ka to var vienkārši pielāgot jebkuram projektam.

Zend piedāvā savu serveri. Uzreiz sakomplektēts serveris ar PHP, domāts tieši projektu izstrādi ZF2 ietvarā.

Zend dod garantiju apkalpot lietotājus par maksu.

2. IZSTRĀDE

Tika izstrādāta datu bāze un divas tīmekļa vietnes. Tīmekļa vietņu izveidošanas mērķis ir vienādu uzdevumu veidošana, divos ietvaros, lai tos varētu salīdzināt.

Sistēmas galvenās funkcija:.

- Lietotāju reģistrācija.

- Lietotāju autorizācija.
- Kategoriju izveidošana un dzēšana.
- Uzdevumu veidošana un dzēšana.
- Uzdevumu redaktēšana.

Papildus sistēmas funkcijas:

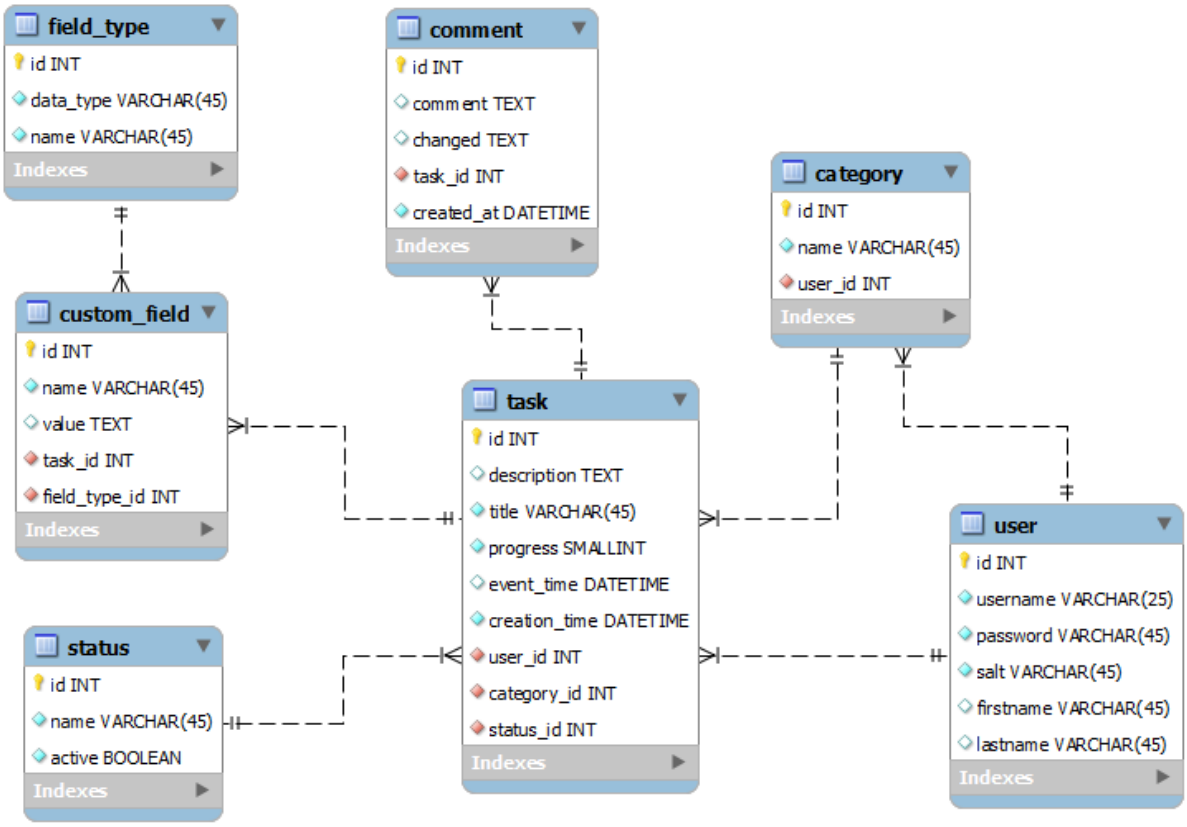
- Komentāru pievienošana uzdevumam.
- Redaktēšana un izmaiņu pierakstīšana.
- Papildus lauku veidošana.

Symfony2 ietvarā tika realizēta pilnā sistēma, kurā ietilpst visas galvenās sistēmas funkcijas un arī visas papildus funkcijas. ZF2 ietvarā tika realizētas tikai galvenās sistēmas funkcijas.

Symfony2 ietvarā veidota sistēma ir pieejama tīmeklī, pēc adreses – http://symfony.beinarovic.lv, un ZF2 ietvarā veidotā sistēma ir pieejama, pēc adreses – http://zend.beinarovic.lv. Tās izmanto kopīgu datu bāzi, un reģistrējoties vienā no tīmekļa vietnēm, autorizēties, kā arī pārvaldīt uzdevumus un kategorijas var abos.

2.1. Datu bāze

Tika izstrādāts ER-modelis, balstoties uz kura tika izstrādāta tīmekļa vietne.



Ilustrācija 2.1 ER-Modelis izstrādātai sistēmai, kurā ar zilu atzīmētie lauki ir obligāti, ar sarkanu ārējās atslēgas bet ar atslēgu primārās atslēgas.

2.1.1. Uzdevuma tabula

Šinī tabulā glabājas visi uzdevumi. Šī ir projekta galvenā tabula, kurai ir ārējās atslēgas uz lietotāju, kategoriju un statusu.

Tabula 2.1 Tabula "task".

Lauks	Datu tips	Apraksts
id	Integer	Automātiski ģenerējošā primārā atslēga.
title	Varchar	Uzdevuma nosaukums.
description	Text	Uzdevuma apraksts.
progress	Small integer	Lauks kurš norāda uz cik procentiem ir izpildīts uzdevums.
event_time	Datetime	Uzdevuma realizācijas brīdis.
creation_time	Datetime	Uzdevuma iveidošanas brīdis.

user_id	Integer	Ārējā atslēga uz lietotāja tabulu.
category_id	Integer	Ārējā atslēga uz kategorijas tabulu.
status_id	Integer	Ārējā atslēga uz uzdevuma stāvokļa tabulu.

2.1.2. Lietotāja tabula

Šinī tabulā glabājas lietotāji, tā tiek izmantota autorizācijai.

Tabula 2.2 Tabula "user".

Nosaukums	Datu tips	Apraksts
id	Integer	Automātiski ģenerējošā primārā atslēga.
username	Varchar	Lietotāja izvēlētais vārds, kuru izmanto lai identificētu lietotāju pie autorizācijas. (unikāls)
password	Varchar	Parole shifrētā veidā.
salt	Varchar	Šifrētā virkne, ar kuras palīdzību tiek iekodēta parole.
firstname	Varchar	Lietotāja vārds.
lastname	Varchar	Lietotāja uzvārds.

2.1.3. Kategrijas tabula

Šinī tabulā glabājas kategorijas. Katram lietotājam tās ir savas. Tās var pievienot un izdzēst, un tām var pievienot uzdevumus.

Tabula 2.3 Tabulsa "category".

Nosaukums	Datu tips	Apraksts
id	Integer	Automātiski ģenerējošā primārā atslēga.
name	Varchar	Kategorijas nosaukums.
user_id	Integer	Ārējā atslēga uz lietotāja tabuli.

2.1.4. Statusu tabula

Tabulā glabājas konstantie uzdevumu statusi.

Tabula 2.4 Tabula "status".

Nosaukums	Datu tips	Apraksts
id	Integer	Automātiski ģenerējošā primārā atslēga.
name	Varchar	Statusa nosaukums. (unikāls)
active	Boolean	Karogs, atzīmē vai nu uzdevums ar šo statsu vēl ir aktīvs.

```
INSERT INTO `todo`.`status` (`name`, `active`)
VALUES ('New', 1),
('In progress', 1),
('Paused', 0),
('Closed', 0);
```

Ilustrācija 2.2 SQL pieprasījums, kurš ievieto konstantās vērtības statusu tabulā.

2.1.5. Komentāru tabula.

Šinī tabulā glabājas gan komentāri, gan veiktās izmaiņas.

Tabula 2.5 Tabula "comment".

Nosaukums	Datu tips	Apraksts
id	Integer	Automātiski ģenerējošā primārā atslēga.
comment	Text	Lietotāja ievadītais komentāra teksts.
changed	Text	Automātiski ģenerēts teksts, kurā aprakstītas veiktās izmaiņas.
created_at	Datetime	Izveidošanas brīdis.
task_id	Integer	Ārējā atslēga uz uzdevuma tabulu.

2.1.6. Datu tipa tabula

Šinī tipā glabājas konstantie pieejamie datu tipi. Šie tipi izmantojas lai pievienotu papildus laukus uzdevumam.

Tabula 2.6 Tabula "field_type".

Nosaukums	Datu tips	Apraksts
id	Integer	Automātiski ģenerējošā primārā atslēga.
data_type	Varchar	Datu tips.
name	Varchar	Datu tipa nosaukums pieejamais lietotājiem.

```
INSERT INTO `todo`.`field_type` (`data_type`, `name`)
VALUES ('TEXT', 'Text'),
('DATETIME', 'Date Time');
```

Ilustrācija 2.3 SQL pieprasījums kurš ievieto konstantās vērtības datu tipa tabulā.

2.1.7. Papildus lauka tabula

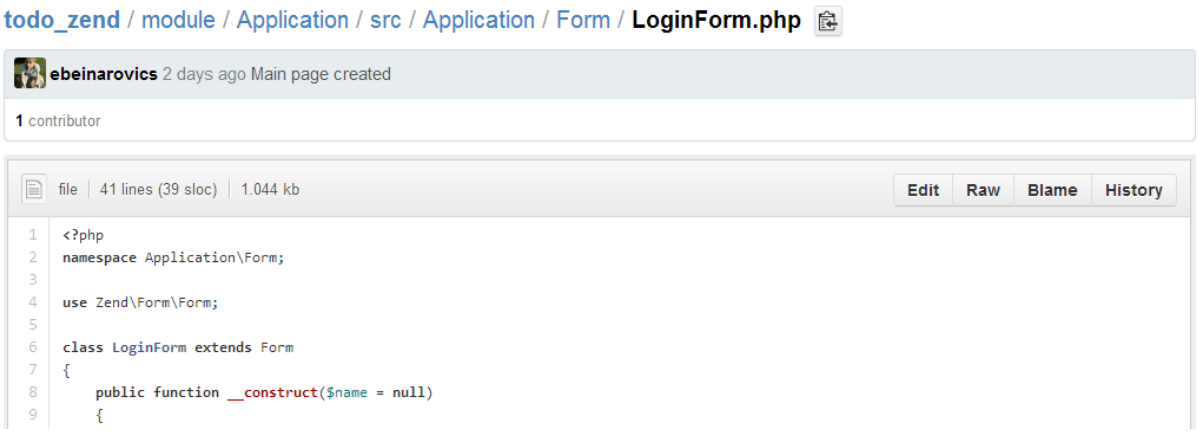
Šinī tabulā glabājas papildus lauki uzdevumam. Papildus lauks pievienojas uzdevumam un tam ir jāuzrāda datu tips.

Tabula 2.7 Tabula "custom_field".

Nosaukums	Datu tips	Apraksts
id	Integer	Automātiski ģenerējošā primārā atslēga.
name	Varchar	Papildus lauka nosaukums.
value	Text	Papildus lauka vērtība. Teksta formā, lai tajā varētu ievietot jebko.
task_id	Integer	Ārējā atslēga uz uzdevuma tabulu.
field_type_id	Integer	Ārējā atslēga uz datu tipa tabulu.

2.2. Versiju kontrole

Versiju kontrolei tika pielietots rīks „Git". Tika izveidoti divi repozitoriji, „GitHub" sistēmā, katram no projektiem. Versiju kontrole ir svarīga lai varētu atgriezties jebkuram saglabātam izstrādes brīdim. Kā arī „Git" dod iespēju novērot visa projekta datnes tīmeklī.



Ilustrācija 2.4 Datnes pārskats "GitHub" tīmekļa vietnes vidē.

2.3. Symfony2 ietvars

Sākotnēji tika izstrādāta tīmekļa vietne Symfony2 ietvara vidē. Symfony2 ir nokomplektēts no dažām no labākajām komponentēm, tīmekļa vietņu izstrādei. Tādām kā Doctrine2 un TWIG.

2.3.1. Instalēšana

Symfony2 sākotnēji vajag lejuplādēt no viņu oficiālās mājas lapas. Ietaicams visjaunāko versiju, jo viņiem pastāvīgi parādās atjauninājumi, ar uzlabojumiem.

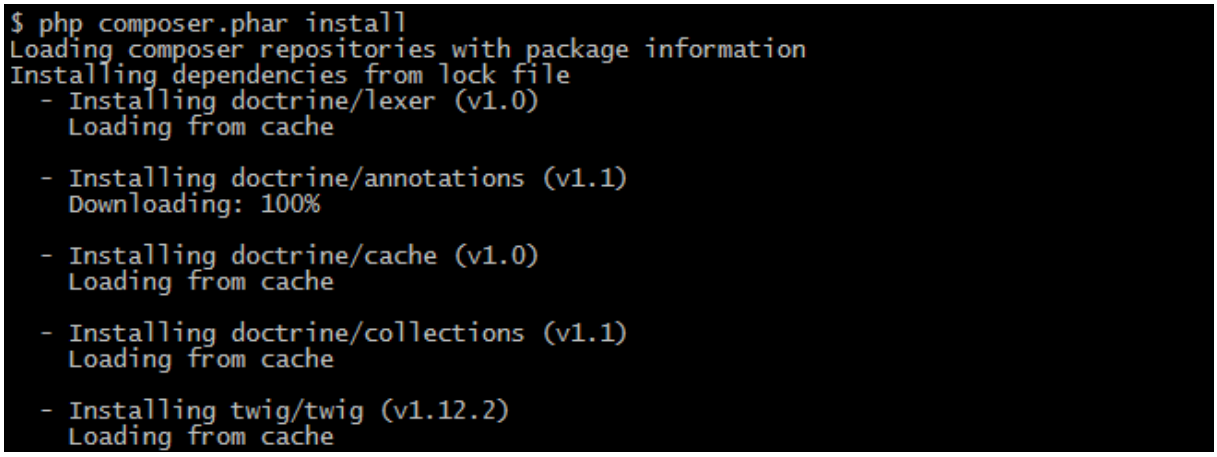
Instalācija notiek izmantojot "Composer" rīku, kurš apskatās visas komponentes, norādītas rīka uzstādījumu datnē, un lejuplādē tās. Sākotnēji ir manuāli jalejuplādē šo rīku, jo Symfony2 paketē ir tikai viņa uzstādījumi.

Symfony2.2 sastāv no 10 komponentēm.

Tabula 2.8 Sākotnējās Symfony2.2 komponentes.

Identifikātors	Nosaukums	Versija

"symfony/symfony"	Symfony2 pakete.	2.2
"doctrine/orm":	Doktrine2.	2.2.3
"doctrine/doctrine-bundle"	Doktrine2 pārvaldības pakete.	1.2
"twig/extensions"	Twig valodas bibliotēka, ar paplašinājumiem priekš Symfony2 ietvara.	1.0
"symfony/assetic-bundle"	Assetu pārvaldības pakete.	2.1
"symfony/swiftmailer-bundle"	E-pastu pārvaldības pakete.	2.2
"symfony/monolog-bundle"	Logošanas pakete.	2.2
"sensio/distribution-bundle"	Pakešu kontroles pakete.	2.2
"sensio/framework-extra-bundle"	Anotāciju parvaldības pakete.	2.2
"sensio/generator-bundle"	Pakete koda ģenerēšanai.	2.2

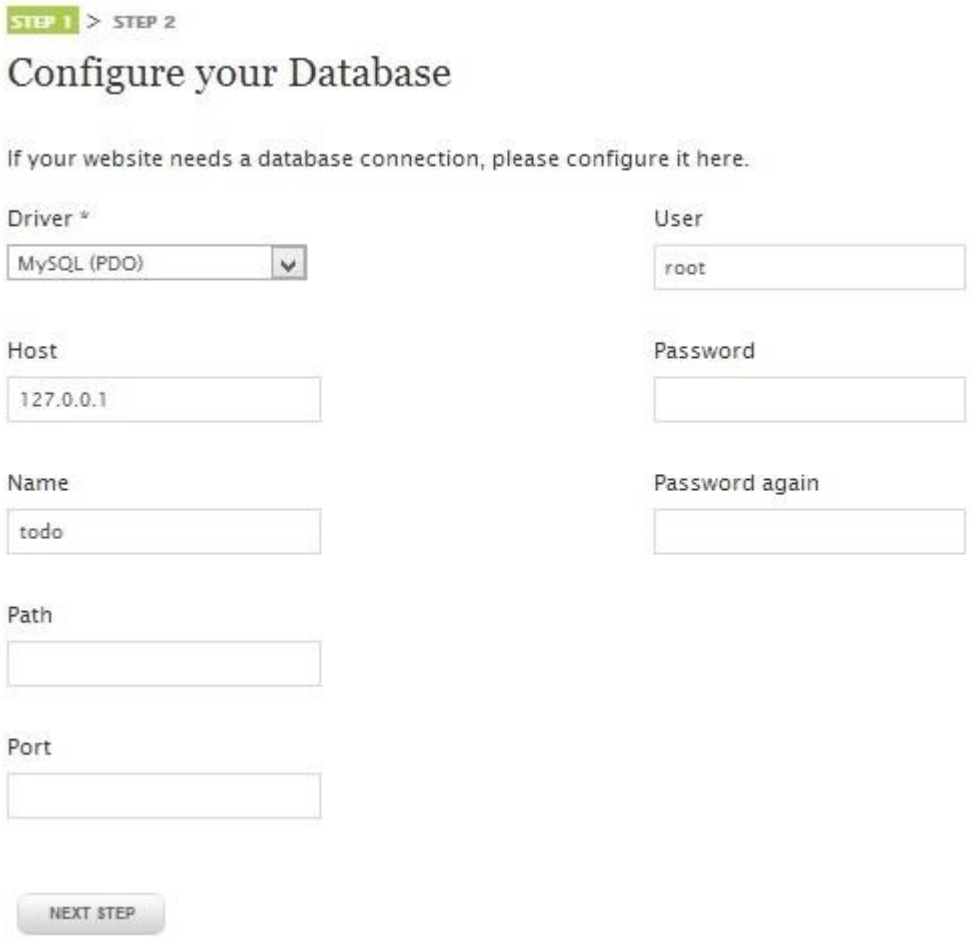


Ilustrācija 2.5 "Composer" lejuplādē visas komponentes. Arī iekļaujot 2. līmeņa komponentes.

Pieinstalējot Symfony2, tas piedāvā konfigurēt datu bāzi caur tīmekļa interfeisu. Tiek piedāvāti vairāki datu bāzes veidi:

- MySQL (PDO)
- SQLite (PDO)
- PosgreSQL (PDO)
- Oracle (native)
- IBM DB2 (native)

- Oracle (PDO)
- IBM DB2 (PDO)
- SQLServer (PDO)



Ilustrācija 2.6 Iebūvētais datu bāzes uzstādīšanas rīks tīmekļa formā.

Kā arī to pašu var panākt pierakstot uzstādījumus „parameters.yml" datnē.

Sākotnēji visi uzstādījumi Symfony2 projektam ir YAML formātā. Eksistē „config.yml" datne kurā ir galvenie publiskie uzstādījumi, kā arī maršrutēšanai ir „routing.yml" datnē.

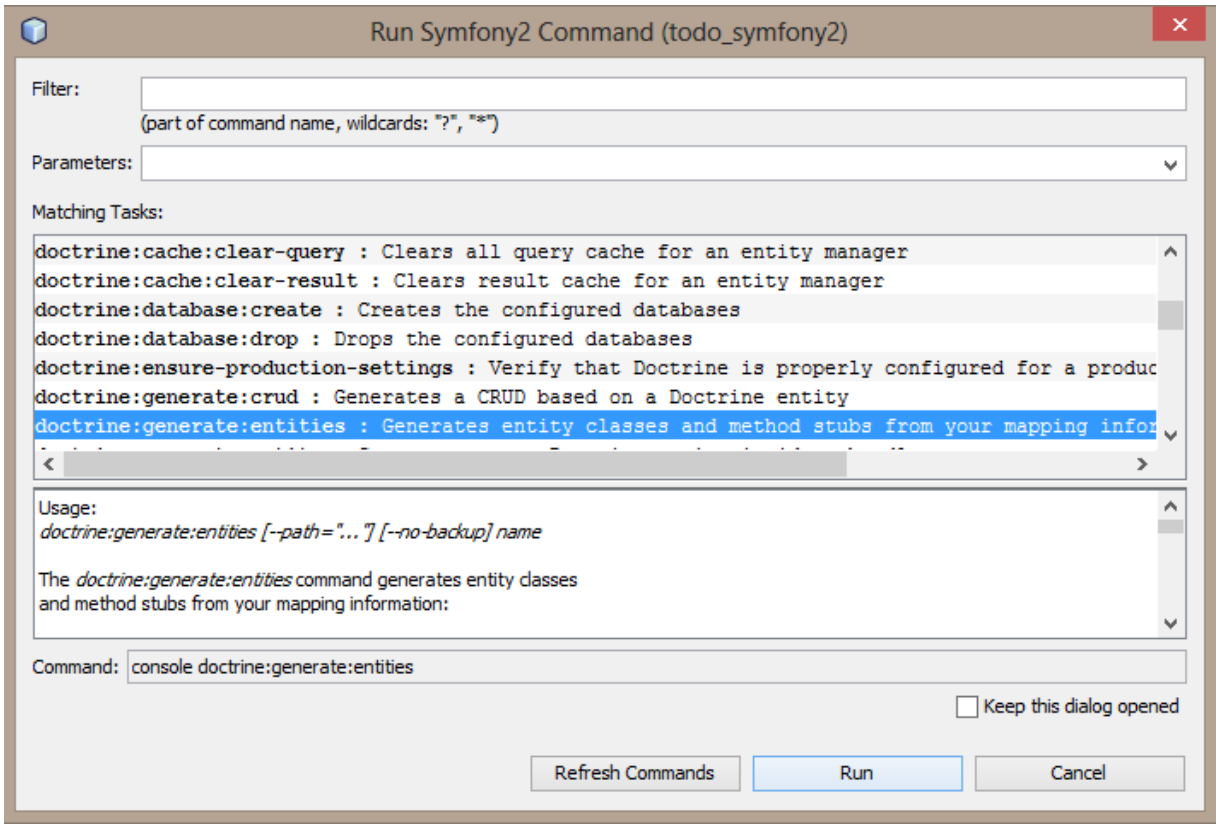
Privātos uzstādījumus, kā piemēram paroles un ceļus līdz datnēm, glabā privātajā failā „parameters.yml". Parasti pielietojot kādu versiju kontroli, šo datni nepievieno un katram lietotājam tā ir unikāla.

Visus uzstādījumus kas var būt YAML formātā, var aizvietot ar XML formātu.

2.3.2. Vienkāršā izstrāde

Balstoties uz dokumentāciju nav grūti sākt darbu un izveidot kādus skatus.

Priekš Symfony2 ir ieteicams lietot izstrādes programmatūru „NetBeans", tā ļoti labi darbojas ar objektorientēto PHP, kā arī tai ir speciāla pievienojumprogrammatūra priekš Symfony2. Tā dod iespēju palaist komandas, veidot jaunus Symfony2 projektus kā arī ļauj labi strādāt ar TWIG programmēšanas valodu.



Ilustrācija 2.7 "NetBeans" programmatūras mijiedarbība ar Symfony2. Komandu saraksts, no kura var palaist komandas.

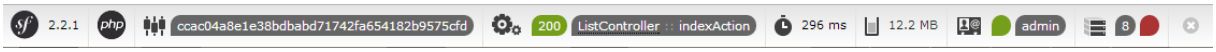
Symfony2 ir paka priekš autorizācijas. Nepieciešams izveidot autorizācijas maršrutus un autorizācijas veidlapu. Uzstādījumu datnē jānorāda no kurienes jāņem lietotājus un kāds ir šifrēšanas algoritms.

2.3.3. Izstrādes vide

Symfony2 piedāvā divas datnes pieprasījuma pieņemšanai – „app.php" un „app_dev.php". Kur „app_dev.php" ir domāts izstrādes gaitam. Veicot pieprasījumus uz „app_dev.php" tiks pieslēgti vairāki moduļi kuri pārķers kļūdas un saglabās visus pieprasījumu datus. Uzstādījumus var atsevišķi uzstādīt izstrādes videi, kas var ļoti bieži

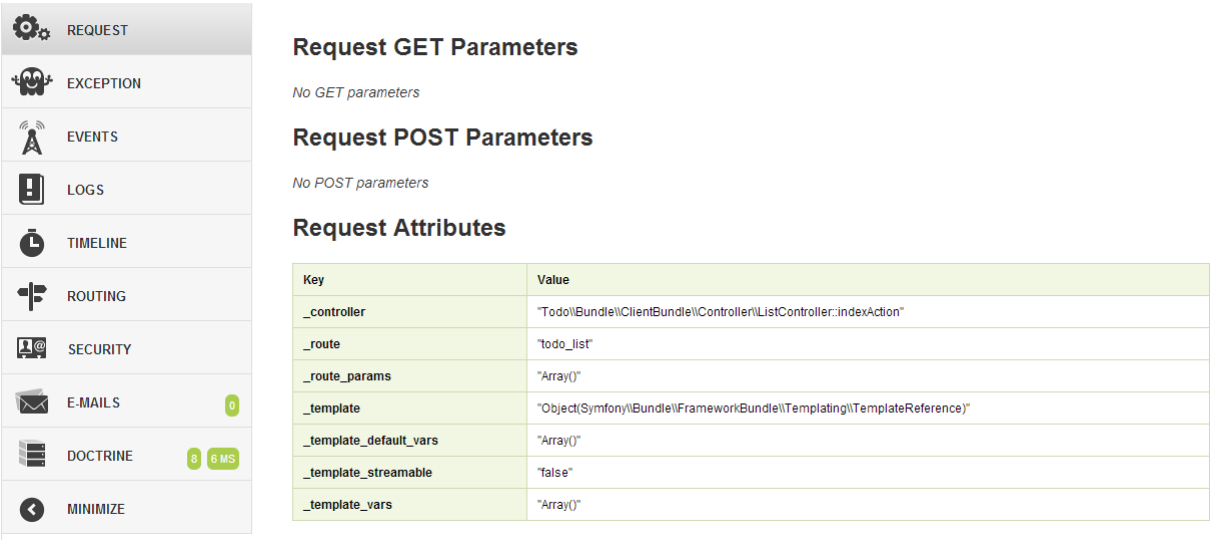
noderēt, jo visbiežāk izstrādā uz vietējā servera, kuram var būt savi uzstādījumi, vai nu kādas paketes nav vajadzības ieslēgt.

Kā arī parādīsies izstrādes plakne, kurā attēlosies būtiska informācija par pieprasījumu.



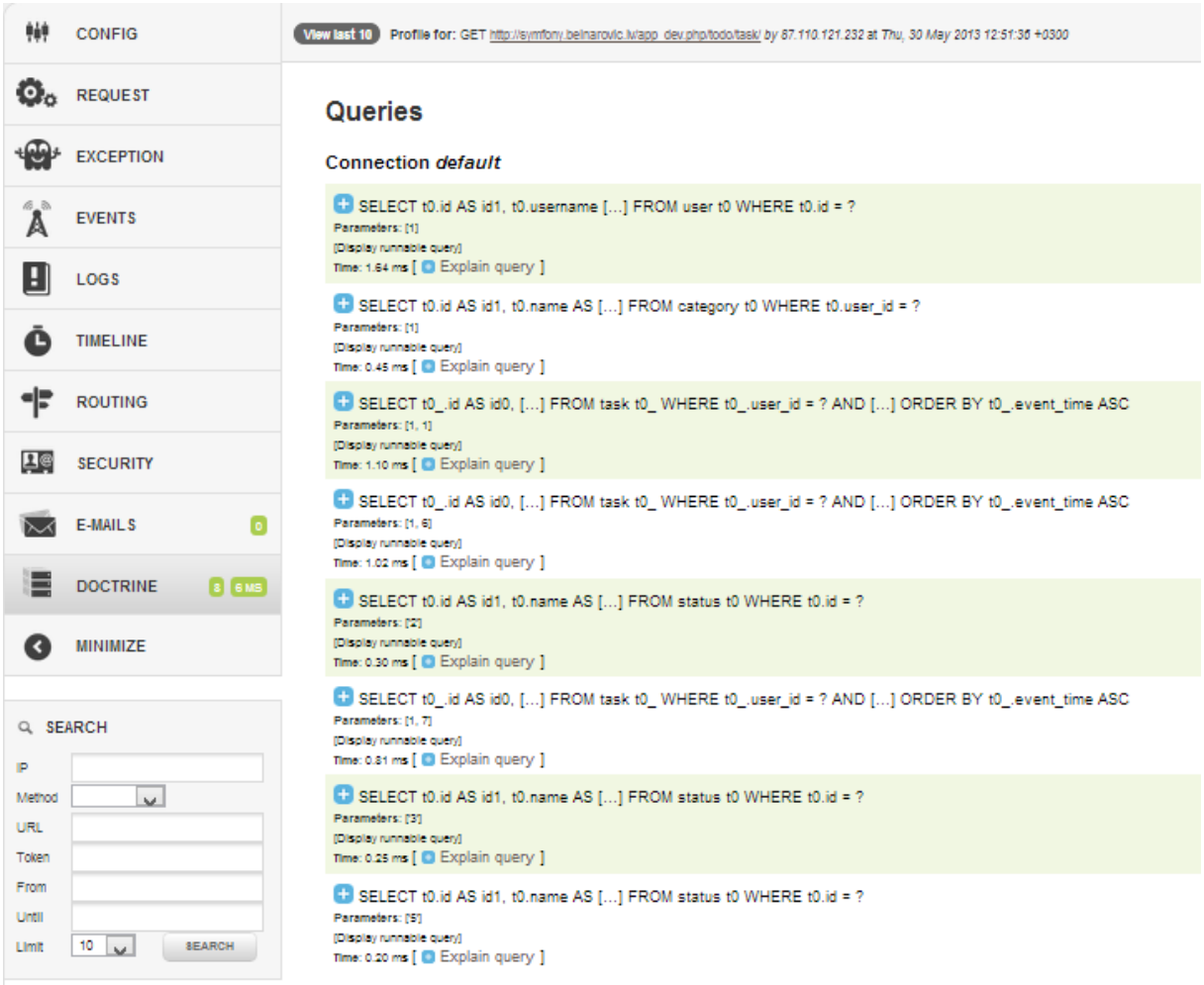
Ilustrācija 2.8 Izstrādes plakne izstrādes vidē.

To paveic rīks „profiler", kurš seko visiem pieprasījumiem veiktiem serverī. Saglabājot visus veiktos pieprasījums, smalki pierakstot visus datus.



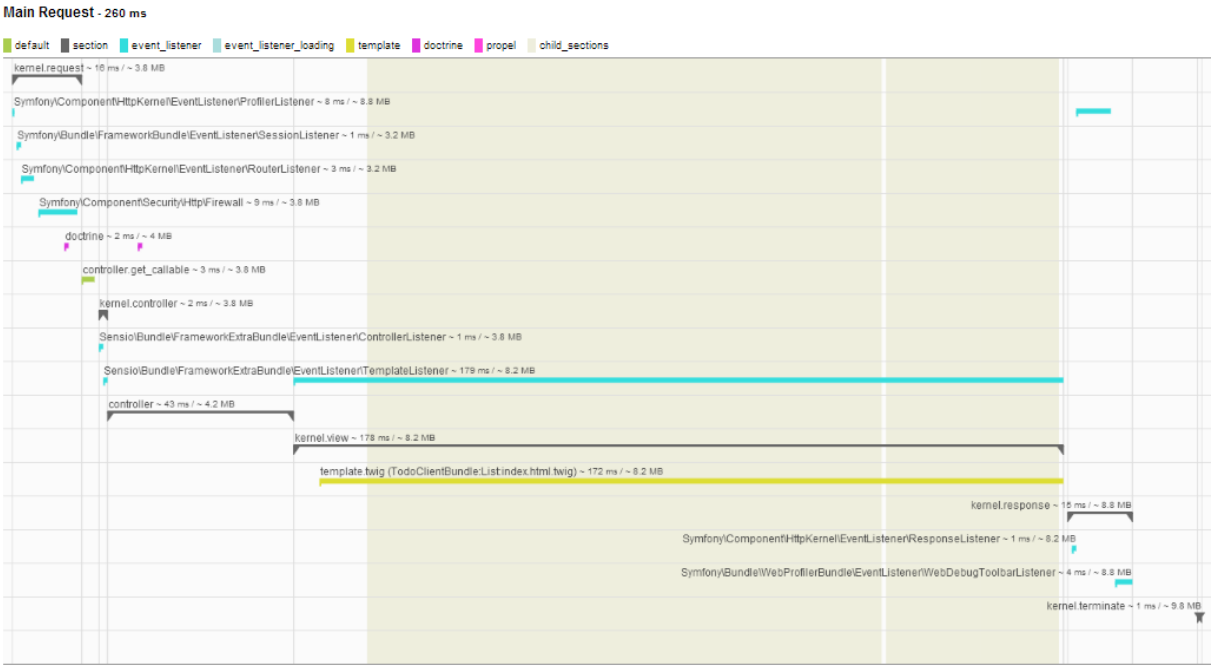
Ilustrācija 2.9 Pieprasījuma pārskats izstrādes vidē.

Kā arī tiek pierakstīti dati par datu bāzes pieprasījumiem. Tiek pierakstīts pats pieprasījums, kā arī tā izpildes laiks.



Ilustrācija 2.10 Datu bāzes pieprasījumu pārskats izstrādes vidē.

Tajā ir meklētājs kurš ļauj meklēt notikumus žurnālā, pēc vairākiem kritērijiem. Kopā tas viss var ietaupīt ļoti daudz laika.



Ilustrācija 2.11 Šinī laika plūsmas grafikā var redzēt pieprasījuma apstrādi.

Vāju vietu meklēšanai, ļoti noderīgs var būt laika plūsmu grafiks. Šinī piemērā ir redzams ka ļoti daudz laika aizgāja uz skata apstrādi. Toties visām apstrādēm jābūt ārpus skata. Uzreiz var saprast ko vajadzētu pielabot.

2.3.3.1. Maršrutēšana

Maršrutēšanas uzstādījumi ir realizēti YAML vidē. Maršrutēšanai ir atvēlēta datne „routing.yml”, tajā ir paredzēts ierakstīt galvenos maršrutus uz atsevišķām projekta paketēm. Piemēram, uzdevumu saraksta paketei URI prefiks būs „/list”, bet kategoriju paketei būs „/category”. Kā arī jābūt uzrādītam ceļam uz nākamo maršrutēšanas datni, ja šī nav pēdējā.

Kā arī ir iespēja padot tālāku maršrutēšanu kontrolieriem, kuri atrodas norādītajā paketē. Tādā gadijumā maršrutēšanas uzstādījumiem jābūt anotāciju formā.

```
todo_security:
    resource: "@TodoSecurityBundle/Controller/"
    type:     annotation
    prefix:   /

todo_client:
    resource: "@TodoClientBundle/Controller/"
    type:     annotation
    prefix:   /
```

Ilustrācija 2.12 Maršrutēšanas piemērs pielietojot anotācijas. Šinī piemērā tiek padota tālākā maršrutēšana uz kontrolieru anotāciju uzstādījumiem.

2.3.3.2. Anotācijas

Sākot ar Symfony2.2, ietvars sākotnēji piedāvā izmantot anotāciju uzstādījumus. Kontrolieru maršrutēšanas uzstādījumi, datu bāzes klašu uzstādījumi, injekcijas un citi, tiek piedāvāti anotāciju veidā.

```
/**
 * @Route("/create", name="todo_task_create")
 * @Template()
 */
public function createAction()
{
```

Ilustrācija 2.13 Metodes maršrutēšanas piemērs kontrolierī. Kur anotācija Route norāda uz URI segmentu šai metodei un "todo_task_create" ir identifikātors. Template anotācijas uzstādījums nodrošinās atbilstošā skata attēlošanu.

```
/**
 * @var \DateTime
 *
 * @ORM\Column(name="creation_time", type="datetime", nullable=false)
 */
private $creationTime;

/**
 * @var \User
 *
 * @ORM\ManyToOne(targetEntity="User")
 * @ORM\JoinColumns({
 *   @ORM\JoinColumn(name="user_id", referencedColumnName="id")
 * })
 */
private $user;
```

Ilustrācija 2.14 Piemērs lauku uzstādījumiem ar anotācijām. Kas norāda kurā mainīgajā tiks injektēta kāda vērtība, kā arī kāds tas ir lauks datu bāzē. Lietotāja gadījumā tiek norādīta tabulu attiecība.

2.3.3.3. Programmēšanas valoda TWIG

Skatu veidošanai Symfony2 pielieto programmēšanas valodu TWIG. Šai valodai piemīt minimalizma specifika.

```
{% extends 'TodoClientBundle::layout.html.twig' %}

{% block content %}
    <p class="text-center">
        <a href="{{ path('todo_task_create') }}" class="btn btn-success">New Task</a>
        <a href="{{ path('todo_category_create') }}" class="btn btn-success">New Category</a>
    </p>
    <div class="row-fluid">
        <div class="span12">
            <h2>Dashboard</h2>
            {% for category in categories %}
                <h4>{{ category }}</h4>
                {% render controller('TodoClientBundle:List:tasksForCategory', { 'category': category }) %}
            {% endfor %}
        </div>
    </div>
{% endblock content %}
```

Ilustrācija 2.15 TWIG programmēšanas valodas piemērs.

Tas ir domāts tieši skatiem un ir saspiests analogs PHP programmēšanas valodai. Tā dod iespēju izpildīt standartos PHP priekšrakstus, izvadīt mainīgo vērtības, apstrādāt tās.

Tajā ir tikai divas birkas – „{{ ... }}" un „{% ... %}".

- {{ ... }} – Izvada vērtību. Tajā var arī būt kādi filtri vai funkcijas, kuri gala rezultātā izvada vērtību.
- {% ... %} – Izpilda darbību.

Symfony2 izveidoja savu pievienojamprogrammu priekš TWIG valodas. Tajā ietilpst vairāki filtri un funkcijas, kā piemēram funkcija - „path". „path" ģenerē un atgriež relatīvo URL un norādīto maršrutu, kā arī dod iespēju izvadīt kāda cita kontroliera metodes resultātu. Pirms mainīgiem nav jāuzrāda uz to, ka tas ir mainīgais.

TWIG piedāvā izmantot blokus un izkārtojuma datnes lai nebublētu kodu.

Assetus pievieno ar attiecīgām birkām, JavaScript datnes ar „javascripts" un CSS datnes ar „stylesheets". Var uzrādīt filtru „Yui Compress", lai visas vienāda tipa datnes katram atsevišķam skatam savienotos 1. Tas dod priekšroku pārlūkprogrammai nolasot to.

```
{% block javascripts %}
    {{ parent() }}
    {% javascripts
        "@TodoClientBundle/Resources/public/js/jquery-1.9.1.js"
        "@TodoClientBundle/Resources/public/js/jquery-ui-1.10.3.custom.min.js"
        "@TodoClientBundle/Resources/public/js/bootstrap.js"
        "@TodoClientBundle/Resources/public/js/time-picker-extension.js"
    %}
        <script src="{{ asset_url }}"></script>
    {% endjavascripts %}
{% endblock javascripts %}

{% block body %}
    <div class="navbar">
        {% include 'TodoClientBundle:Include:header.html.twig' %}
    </div>

    <div class="container">
        {% block content %}{% endblock content %}
        <div class="row-fluid">
```

Ilustrācija 2.16 Izkārtojuma datnes piemērs. Tiek parādīts kā apzīmē "content" bloka novietojumu, „javascripts" blokā tiek attēlota assetu pieslēgšana.

Tas ļoti atvieglina darbu ar skatiem, jo pielietojot PHP skatos, aiziet daudz laika uz PHP birku atvēršanu un aizvēršani, kartam mainīgam jāliek „$" simbols priekšā, kā arī bloku sistēma ir ļoti vienkārša.

2.3.4. Saskarsme ar datu bāzi

Symfony2 izmanto Doctrine2, darbam ar datu bāzi. Doctrine2 ir ļoti atzīta ORM sistēma, darbam ar datu bāzi. Šīm sistēmām ir ļoti laba mijiedarbība jo Symfony2 ir balstīts uz Doctrine2. Nav napieciešamības neko papildus konfigurēt.

2.3.4.1. Doctrine2

Doktrīnei ir bagāta dokumentācija viņu mājaslapā.

Kā katram ORM rīkam, Doctrine2 arī pieprasa kartēšanu. Jābūt izveidotām klasēm ar mainīgajiem. Klasei jābūt uzrādīts ka tā ir „Entity" un jānorāda uz kādu tabulu tā attiecas, katram mainīgajam ir jāuzrāda uz kādu lauku, tabulā, tas attiecas. Kā arī ir nepieciešamas uzstādīšanas un nolasīšanas metodes, katram laukam, nolasīšanas metodēm nosaukuman jābūt tādam pašam kā mainīgajam, bet ar prefiksu „get" un camelcase formātā Uzstādīšanas metodēm jābūt prefiks set vai add. Kārtēšanā var arī uzstādīt ārējās atslēgas uz citām tabulām.

```
/**
 * @ORM\ManyToOne(targetEntity="Task", inversedBy="customFields")
 * @ORM\JoinColumns({
 *   @ORM\JoinColumn(name="task_id", referencedColumnName="id")
 * })
 */
private $task;
```

Ilustrācija 2.17 Kartēšanas piemērs, anotāciju vidē, no papildus lauka klases. Tajā ir norādīta ārējā atslēga uz uzdevuma tabulu. Tiek norādīts ka saite savieno ar „Task" klasi, un ka tajā to atspoguļos „customField" mainīgais.

```
/**
 * @ORM\OneToMany(mappedBy="task", targetEntity="CustomField", cascade={"persist", "remove"})
 * @ORM\JoinColumns({
 *   @ORM\JoinColumn(referencedColumnName="id")
 * })
 */
private $customField;
```

Ilustrācija 2.18 Kartēšanas piemērs, anotāciju vidē, no uzdevuma klases. Tiek norādīts kurš lauks to atspoguļos. Tas ir nepieciešams lai varētu pa taisno no šī objekta dabūt papildus lauka objektus, izmantojot nolasīšanas metodi „getCustomField". Kā arī ir pievienoti cascade trigeri. Šie ustādījumi van obligāti.

Doctrine2 pie katra „flush" metodes pieprasījuma sinhronizē visus tai pazīstamos objektus. Lai Doctrine2 zinātu kurus objektus ir jāsinhronizē, ar datu bāzi, uz tiem objektiem jāizsauc metodi „persist". Uz nolasītiem no datu bāzes objektiem tas neattiecas, jo viņi jau ir savienoti ar Doctrine2. Lai Doctrine2 nereaģētu uz objekta izmaiņām, var izsaukt metodi „detach", šinī gadījumā Doctrine2 neskatīsies vai tas ir mainījies un vai vajag to sinhronizēt.

Pie katra „flush" pieprasījuma, Dontrine2 pārbauda visus tai zināmos objektus, to izmaiņas un sagatavo SQL pieprasījumu. Nav nepieciešamības pievērst uzmanību uz secību, jo Doctrine2 izskaidro kuriem pieprasījumiem jābūt sākumā, bet kurem vēlāk. Tas ir ļoti noderīgi ar ārējām atslēgām.

```
$em = $this->getDoctrine()->getEntityManager();

$em->persist($user);

$category = new Category();
$category->setUser($user);
$category->setName('ToDo');

$em->persist($category);

$em->flush();
```

Ilustrācija 2.19 Piemērs kā ierakstās datu bāzē vienlaikus lietotājs un kategorija. Ierakstīšana datu bāzē notiek "flush" metodes izsaukšanas brīdī.

Ir iespēja ģenerēt nepieciešamās klases, un no tām ģenerēt datu bāzi. Kā arī kā šinī darbā tika darīts – var no jau esošās datu bāzes izveidot nepieciešamās priekš Doctrine2 klases. Paveikt to patstāvīgi aizņemtu daudz laika, jo katram laukam jābūt mainīgais ar uzstādījumiem, kā arī saņemšanas un uzstādīšanas metodes. Datu bāzei paredzētai šim pētījumam tas varētu aizņemt vairāk par stundu. Balstoties uz oficiālo dokumentāciju, tas tika paveikts 10 minūšu laikā. To var paveikt palaižot konsoles komandu „doctrine:mapping:convert” un norādot uzstādījumu „—from-database”, vietu kur izveidot datnes ar uzstādījumiem un vēlamo uzstādījumu formātu.

Tika izveidoti uzstādījumi „XML” formātā, uzrādītajā TodoCoreBundle paketē.

Lai uzģenerēt entitiju klases šiem uzstādījumiem ir komanda „doctrine:import:mapping”. Tai jānorāda paķeti kurā atrodas uzstādījumi un kurā tiks izveidotas klases, kā arī klašu konfigurāciju formātu. Tika izvēlētas anotācijas kā uzstādījumu formāts. Atliek vien palaist komandu „generate:doctrine:entities”, kura izveidos metodes vērtību nolasīšanai un uzstādīšanai.

2.3.4.2. Repozitoriji

Grūtākiem datu bāzes pieprasījumiem Symfony2 piedāvā izmantot repozitorijus. Repozitoriji ir klases kurās tiek veidoti un izsaukti DQL pieprasījumi.

```
class TaskRepository extends EntityRepository
{
    public function getByUserAndCategory(User $user, Category $category)
    {
        return $this->createQueryBuilder('t')
                ->where('t.user = :user')
                ->andWhere('t.category = :category')
                ->orderBy('t.eventTime')
                ->setParameters(array(
                    'category'  => $category,
                    'user'      => $user
                ))
                ->getQuery()
                ->getResult();
    }
}
```

Ilustrācija 2.20 Lietotāju repozitorija piemērs. Piemērā pieprasījums tiek veidots ar pieprasījumu būvētāja klasi.

Ir divi veidi kā veidot DQL pieprasījumus, var rakstīt parastu DQL pieprasījumu. Tas galvenokārt atšķirās no SQL pieprasījumiem ar to, ka tiek automātiski savienotas tabulas pēc uzrādītām attiecībām, nav jāraksta „Join" nosacījums. Kā arī Doctrine2 piedāvā pieprasījumu būvētāja klasi. Šī klase piedāvā veidot pieprasījumu objektorientētā PHP vidē.

2.3.4.3. Cascade trigeri

Doctrine2 dod iespēju kartēšanā uzstādīt cascade trigerus. Krietni atvieglina darbu, jo var pierakstīt „cascade remove" trigeri uz saiti, tad izdzēšot šo ierakstu, Doctrine2 izdzēsīs arī attiecīgos ierakstus pēc šīs saites. Kā arī ir tāds trigeris priekš „persist", kas nozīmē, ka tikko izsaucot metodi persist uz šī objekta, tas automātiski izsauksies uz attiecīgiem objektiem, pēc saites.

2.3.5. Veidlapas un validācija

Veidlapas ir ļoti ērti realizētas, tās balstās uz doktrīnes klasēm. Tiek piedāvāts ģenerēt veidlapas uzstādījumu klasi no Doctrine2 objektu klasēm, izmantojot konsoles komandas. Veidlapu uzstādījumu klasēs.

Veidlapu validācija notiek automātiski, nepieciešams pievienot lauku validācijas nosacījumus. Validācijas nosacījumus var definēt PHP vidē, anotāciju vidē, YAML vidē vai nu XML vidē. Pētījumam tika izmantota anotāciju vide.

```
/**
 * @var string
 *
 * @ORM\Column(name="username", type="string", length=25, nullable=false)
 * @Assert\NotBlank()
 * @Assert\MinLength(
 *     limit=4,
 *     message="Username must have at least {{ limit }} characters."
 * )
 */
private $username;
```

Ilustrācija 2.21 Lauka validācijas nosacījumi, anotāciju vidē. Tiek uzstādīti divi nosacījumi, „NotBlank" – norāda, ka laukam nevar būt tukšam un „MinLength" – norāda minimālo simbolu skaitu – 4, ar savu kļūdas paziņojumu.

Symfony veidlapas klase nodarbojas ar validāciju un veidlapas attēlošanu. Pie izveidošanas tai padodas veidlapas uzstādījumu objekts. „bind" – metode savieno pieprasījumā saņemtos datus ar veidlapas objektu. Tai ir metode „isValid", kura pēc visiem validācijas nosacījumiem, validē veidlapu. Atrodot kļūdas, tā ieraksta tās savus mainīgos, kuri pēc izvēles var būt izvadīti pie veidlapas izvadīšanas skatā.

```
/**
 * @Route("/signup", name="todo_signup")
 * @Method("POST")
 */
public function signupAction()
{
    $request = $this->getRequest();
    $view = 'TodoClientBundle:Homepage:index.html.twig';

    $form = $this->createForm(new UserType());
    $form->bind($request);

    if ($form->isValid()) {
        // ...
    }

    return $this->render($view, array(
        'registration'  => $form->createView()
    ));
}
```

Ilustrācija 2.22 Veidlapas validācijas piemērs.

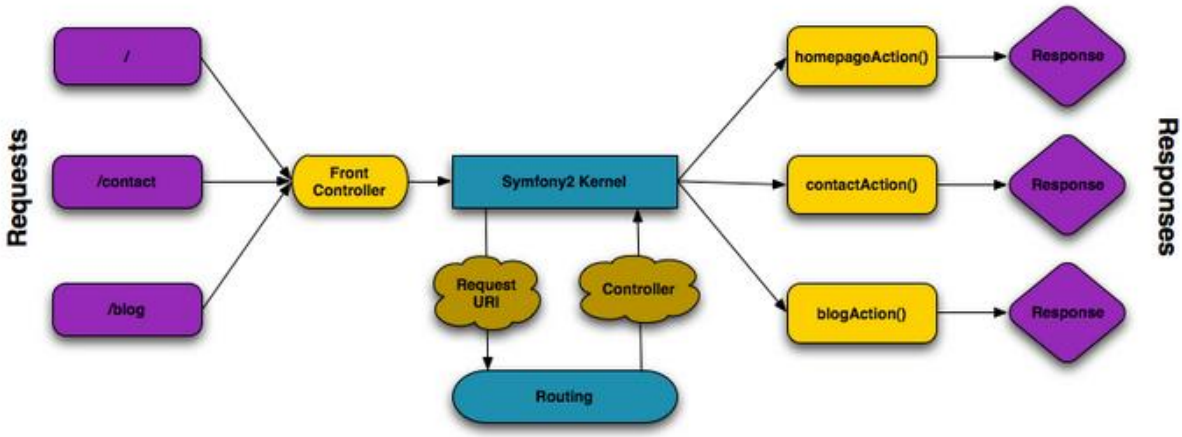
Symfony2 sagatavoja vairākas metodes, TWIG programmēšanas valodai, priekš veidlapām. Var izvadīt uzreiz veselu veidlapu, konkrēta lauka rindu, konkrēta lauka atsevišļo daļu, kā piemēram lauku vai lauka kļūdas, kā arī atsevišķi veidlapas kļūdas.

```
<h3>Registration</h3>
<form action="{{ path('todo_signup') }}"
      method="post" {{ form_enctype(registration) }} autocomplete="off">
    {{ form_widget(registration) }}
    <input type="submit" class="btn btn-primary" value="register">
</form>
```

Ilustrācija 2.23 Veidlapas izvadīšana ar TWIG palīdzību, šinī piemērā izvadās uzreiz vesela forma.

2.3.6. Kodola plūsmas

Katrs pieprasījums tiek apstrādāts Symfony2 kodolā „AppKernel" klasē. Tajā tiek pieslēgtas visas komponentes, klausītāji un kodola servisi. Patstāvīgi tiek veiktas pārbaudes uz izņēmumiem.



Ilustrācija 2.24 Pieprasījumu apstrādes cikls.

Tā kā Symfony2 strādā ar anotācijām, tiek veiktas datņu pārbaudes uz no pieejamību.

2.3.7. Assetic

Tiek pielietota assetic sistēma, kura ģenerē assetus, no ietvara paķetēm uz publisko vietni. Tā mērķis ir filtrēšanā un saspiešana. Nav atkārtoti jāraksta assetu birkas. Izstrādes laikā, katru reizi pēc kāda asseta uzlabošanas, lai apskatīt rezultātu, vajag palaist komandu „assetic:dump". Šī komanda ģenerē visus assetus, priekš skatiem, publiskajā direktorijā. Lai to neatkārtotu, var palaist šo komandu gaidīšanas režīmā. Tad tā pamanot izmaiņas atjaunināsnapiecciešamās datnes.

2.3.8. Komponentes

Šobrīt priekš Symfony2 ir ļoti daudz publiski pieejamo komponentu.

Ļoti daudz komponenšu ir aprakstīti viņu dokumentācijā, kā arī ir ļoti daudz programmētāju, kuri paši izveido kādu komponenti un izliek to publiski tīmeklī. Ir izveidojusies kultūra, kur citi programmētāji uzlabo vairākas komponentes vai uztur tās.

2.3.9. Kešošana

Symfony2 ir iebūvēta kešošana un logošana. Tā kā Symfony2 ir ārkārtīgi kompleksa sistēma, tā strādā ne pārāk ātri. Tieši tādēļ tika izstrādāta speciāla kešošanas sistēma kura krietni paātrina ātrdarbību. Ir ieteikts izmantot arī kādu servera kešošanas rīku kā piemēram „Varnish". Tīmeklī var atrast daudz recenzijas par Symfony2, kur tiek rakstīts, kā ārkārtīgi lieli projekti ar lielu pieprasījumu daudzumu strādā ļoti ātri, ja tiek pielietota servera kešošana.

2.3.10. Konsoles komandas

Symfony2 piedāvā veslelu sarakstu ar komandām kuras izpilda PHP. Tās komandas var krietni paātrināt darbu, izveidojot veselas klases vai nu ģenerējot datu bāzi.

Tabula 2.9 Daļa no piedāvātām komandām Symfony2 ietvarā.

assetic:dump	Ģenerē visas assetu datnes.
assets:install	Pārnes visus asetus no moduļiem uz publisko direktoriju.
cache:clear	Iztīra kešatmiņu.
cache:warmup	Sagatavo kešatmiņu.
container:debug	Uzrāda sarakstu ar pieejamiem servisiem.
doctrine:database:create	Izveido datu bāzi.
doctrine:database:drop	Izdzēš datu bāzi.
doctrine:generate:crud	Izveido kontrolierus pirekš darba ar datu bāzes klasēm. Create, Read, Update un Delete.
doctrine:generate:entities	Izveido uzstādīšanas un nolasīšanas metodes katram laukam datu

	bāzes klasē.
doctrine:generate:entity	Izveido jaunu datu bāzes klasi.
doctrine:generate:form	Izveido veidlapas uzstādījumu klasi priekš datu bāzes klases.
doctrine:schema:update	Izveido tabulas priekš visām datu bāzes klasēm.
generate:bundle	Ģenerē paketi.
generate:controller	Ģēnerē kontroliera klasi.

Kā arī tiek piedāvāts izveidot savas komandu klases. Lai izveidotu savu komandu, pietiek izveidot datni, pēc attiecīga standarta, ar attiecīgu nosaukumu un klasi, mapē „Command", jebkurā paketē. Tā uzreiz reģistrējas ietvara komandās.

```
router
  router:debug                    Displays current routes for an application
  router:dump-apache              Dumps all routes as Apache rewrite rules
  router:match                    Helps debug routes by simulating a path info match
server
  server:run                      Runs PHP built-in web server
swiftmailer
  swiftmailer:spool:send          Sends emails from the spool
todo
  todo:project:author             Shows the author of the project.
translation
  translation:update              Updates the translation file
twig
  twig:lint                       Lints a template and outputs encountered errors
```

Ilustrācija 2.25 Izveidotā "todo:project:author" komanda, komandu sarakstā.

Izsaucot komandu, Symfony2 palaižot to apstrādā ievaddatus, validējot tos.

```
eka-pc@EKA /c/wamp/www/todo_symfony (master)
$ app/console todo:project:author


  [RuntimeException]
  Not enough arguments.


todo:project:author count


eka-pc@EKA /c/wamp/www/todo_symfony (master)
$ app/console todo:project:author 3
The author is Edmunds Beinarovics!
The author is Edmunds Beinarovics!
The author is Edmunds Beinarovics!
```

Ilustrācija 2.26 Pētījumam izveidotās komandas palaišana bez nepieciešamā parametra un ar to.

2.4. Zend Framework 2 ietvars

Šim ietvaram piemīt vienkārša struktūra, tas pilnībā sastāv no PHP programmēsanas valodas, izņemot skatus kuri arī ir PHP vidē bet iekļauj HTML iezīmēšanas valodu un assetus kā „JavaScript" un „CSS".

Zend piedāvā savu serveri kuram ir iebūvēti atkļūdošanas rīki.

2.4.1. Instalēšana

Sākotnēji nepieciešams lejuplādēt projektu no ZF2 oficiālās mājaslapas, kurā tiek pieāvāti vairāki varianti. Var lejuplādēt pulno versiju, ar komponentēm jau pieinstalētām, vai arī parasto versiju kurai nepieciešama komponenšu instalēšana. Kā arī tiek piedāvātas lejuplādēt projektu uzreiz ar Zend serveri.

Instalācijai nepieciešams izmantojot "Composer" rīku, kurš tāpat kā Symfonu2 ietvarā analizē visas komponentes, norādītas komposer uzstādījumu datnē, un lejuplādē tās. Sākotnēji ir manuāli jālejuplādē šo rīku, jo ZF2 lejuplādētā projektā ir tikai tā uzstādījumi.

Pēc instalēšanas uzreiz drīkst veikt pieprasījumus serverim un saņemt parauga datus. Parauga dati, praktiski, ir ļoti ērti. Ir izveidots parauga modulis „Application", ar kontrolieri un skatu. Var uzreiz sākt pārveidot šo moduli sev nepieciešamajam.

Ietvara uzstādījumi glabājas projekta uzstādījumu datnē „application.config.php", uzstādījumu parametri datnē „global.php". Ir atsevišķa datne privāto parametru glabāšanai, „local.php". Maršrutēšana, servisi un kontrolieri tiek uzstādīti katram modulim atsevišķi. Tie glabājas, moduļa, uzstādījumu datnē.

2.4.2. Vienkāršā izstrāde

Priekš šī ietvara ir speciāla programmatūra – „Zend Studio". Balstoties uz aprakstu tā ļoti palīdz izstrādes gaitām, diemžēl viņa ir maksas. ZF2 strādā labi ar „NetBeans", tam ir pievienojumprogrammatūra priekš ZF2 ietvara. Tā dod iespēju veidot jaunus projektus ar ietvara struktūru, kā arī mainīt atvērto datni no metodes uz skatu, un atpakaļ.

Ir daudz variantu kā veikt autorizāciju, bet dokumentācijā tas ir ļoti neskaidri izstāstīts un pienākas daudz ko uzstādīt lai realizēt sev nepieciešamo autorizācijas sistēmu.

Tiek piedāvāts izmantot anotācijas, vairāku darbu paveikšanai.

2.4.2.1. Maršrutēšana

ZF2 ietvarā maršrutēšanas uzstādījumi glabājas PHP programmēšanas valodas mācībā.

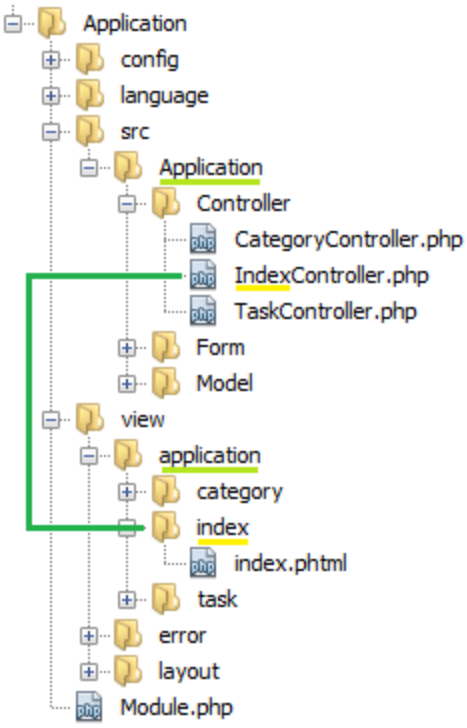
```
return array(
    'router' => array(
        'routes' => array(
            'home' => array(
                'type' => 'Zend\Mvc\Router\Http\Literal',
                'options' => array(
                    'route'    => '/',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Index',
                        'action'     => 'index',
                    ),
                ),
            ),
            'tasks' => array(
                'type' => 'Zend\Mvc\Router\Http\Literal',
                'options' => array(
                    'route'    => '/list',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Task',
                        'action'     => 'index',
                    \
```

Ilustrācija 2.27 ZF2 ietvara maršrutēšanas piemērs.

Tas ir viegli saprotams, jo viss notiek PHP vidē, vienīgi šie uzstādījumi aizņem daudz vietas un var būt ne pārāk saskatāmi.

2.4.3. Skati

Skatu izvadīšana notiek PHP vidē. Skatu izsaukšana no kontroliera ir ļoti ērti realizēta. Ar skatu attēlošanu nodarbojas klase „ViewModel”. Katrai kontroliera metodei var izveidot attiecīgu skatu, un ietvars atradīs to un attēlos.



Ilustrācija 2.28 Kontroliera un skatu attiecība. Tiek attēlots ka "IndexController" kontrolierim atbilst mape "index". Katra metode kontrolierī var automātiski attēlot datni, ar tādu pašu nosaukumu, no tās mapes.

Izkārtojums notiek automātiski, modulim ir standarta izkārtojuma datne, un skats to izmantos. Atsevišķām metodēm var uzstādīt citu izkārtojuma datni.

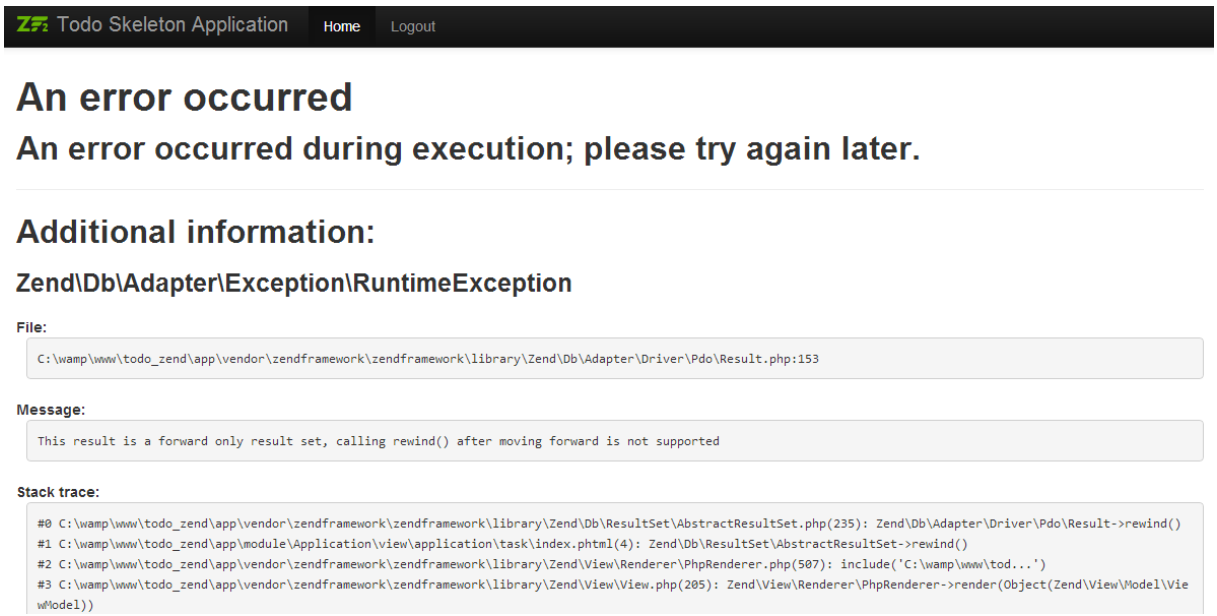
```
public function addAction()
{
    $form = new CategoryForm();
    // ...
    $layout = $this->layout();
    $layout->setTemplate('layout/loggedinLayout');

    return new ViewModel(array(
        'form'      => $form
    ));
}
```

Ilustrācija 2.29 Izkārtošanas datnes uzstādīšana.

2.4.3.1. Atkļūdošana un izņēmumi

Katra moduļa uzstādījumos var uzstādīt standarta skatus, tādus kā izņēmumu situācija un „lapuse nav atrasta". Var arī uzstādīt vai ir vajadzība rādīt iemeslus. Izstrādājot ir ļoti noderīgi lasīt iemeslus, jo tie paskaidro kādēļ šī lapa tika attēlota, un ja ir nepieciešamība, kur to var labot.



Ilustrācija 2.30 Izņēmuma skata, ar paskaidrojumiem, attēlošana pārlūkprogrammā.

2.4.4. Saskarsme ar datu bāzi

ZF2 izmanto savu sistēmu darbam ar datu bāzi. Šī ir ORM sistēma kurai arī ir nepieciešamas klases nolasītiem, no datu bāzes, datiem. Klasei ir jāsatur nepieciešamo lauku mainīgos. Klasei jāsatur metode kura piešķir klases mainīgajiem vērtības no, nolasītā no datu bāzes, masīva. Kā strādā šī sistēma var viegli saprast, tiek norādīta datu bāzes tabula, no kuras dati tiks piešķirti klasei.

```
public function exchangeArray($data)
{
    $this->id           = (isset($data['id']))        ? $data['id']           : null;
    $this->username     = (isset($data['username']))  ? $data['username']     : null;
    $this->password     = (isset($data['password']))  ? $data['password']     : null;
    $this->salt         = (isset($data['salt']))      ? $data['salt']         : null;
    $this->firstname    = (isset($data['firstname'])) ? $data['firstname']    : null;
    $this->lastname     = (isset($data['lastname']))  ? $data['lastname']     : null;
}
```

Ilustrācija 2.31 Datu bāzes klases metode vērtību piešķiršanai.

Toties tas aizņem daudz atkārtojošos koda fragmentus, un definējot šādu metodi tabulai ar daudziem laukiem, var aizņemt daudz lainka lai sarakstīt šīs piešķiršanas. Tika pievienota papildus metode vērtību uzstādīšanai, lai nedublētos mainīgie.

```
    public function exchangeArray($data)
    {
        $this->id           = $this->prepareForAllocation($data, 'id');
        $this->username     = $this->prepareForAllocation($data, 'username');
        $this->password     = $this->prepareForAllocation($data, 'password');
        $this->salt         = $this->prepareForAllocation($data, 'salt');
        $this->firstname    = $this->prepareForAllocation($data, 'firstname');
        $this->lastname     = $this->prepareForAllocation($data, 'lastname');
    }

    private function prepareForAllocation($array, $key)
    {
        return isset($array[$key]) ? $array[$key] : null;
    }
```

Ilustrācija 2.32 Uzlabotā metode vērtību uzstādīšanai, ar papildus metodi.

Būtu labi ja šāds, vai tamlīdzīgs, risinājums būtu automatizēts ietvarā. Jo katrai klasei atkārtot šādu metodi nav pareizi.

2.4.4.1. Tabulu klases

Priekš ierakstu nolasīšanas no datu bāzes, ZF2 izmanto tabulu klases. Tabulu klasē jāpadod „TableGateway" objekts, kurš nodarbojas ar datu bāzi.

```
class UserTable
{
    protected $tableGateway;

    public function __construct(TableGateway $tableGateway)
    {
        $this->tableGateway = $tableGateway;
    }

    public function getByUsername($username)
    {
        $rowset = $this->tableGateway->select(array('username' => $username));
        $row = $rowset->current();
        if (!$row) {
            return false;
        }

        return $row;
    }
    // ...
}
```

Ilustrācija 2.33 Tabulas klases piemērs.

Ir iespējams vieidot sarežģītākus pieprasījumus izmantojot „TableGateway” objektu, ar objektorientētu pieeju.

```
public function customFetchById($id)
{
    $resultSet = $this->tableGateway->select()
            ->from('user')
            ->where('id', $id);

    return $resultSet;
}
```

Ilustrācija 2.34 Objektorientētais pieraksts pieprasījumam izmantojot "TableGateway" klasi.

2.4.5. Veidlapas un validācija

ZF2 priekš veidlapām izmanto klasi „Form”. Veidlapas izveidošanai nepieciešams veidot veidlapu klases, kuras paplašina ZF2 klasi „Form”. Klasei ir jāuzstāda apakšklases „Form” uzstādījumus – kādi ir lauki, ar kādiem uzstādījumiem un ar kādu piekļuves paņēmienu tā tiks sūtīta.

```
class LoginForm extends Form
{
    public function __construct($name = null)
    {
        parent::__construct('user');
        $this->setAttribute('method', 'post');
        $this->add(array(
            'name' => 'username',
            'attributes' => array(
                'type'  => 'text',
            ),
            'options' => array(
                'label' => 'Username',
            ),
        ));
        // ...
    }
}
```

Ilustrācija 2.35 Veidlapas klases piemērs.

Validācijas nosacījumiem jābūt datu bāzes klasē, PHP vidē. Ir iespējams pieslēgt anotācijas, validēšanai. Priekš validācijas filtru ieviešanas, vajag lai datu bāzes klase lieto

(implementē), ZF2 klasi „InputFilterAwareInterface”, un ir jāizveido metode „getInputFilter”, kurai ir jāatgriež uzstādītā validāciju nosacījumu klase.

```
$inputFilter->add($factory->createInput(array(
    'name'     => 'title',
    'required' => true,
    'filters'  => array(
        array('name' => 'StripTags'),
        array('name' => 'StringTrim'),
    ),
    'validators' => array(
        array(
            'name'    => 'StringLength',
            'options' => array(
                'min'      => 4,
                'max'      => 100,
            ),
        ),
    ),
)));
```

Ilustrācija 2.36 Validācijas pievienošanas piemērs PHP vidē.

Veidlapas apstrāde arī notiek ļoti vienkārši. Veidlapas klasei tiek padoti nosacījumi un ar „isValid” metodi ietvars pārbauda visus laukus a validācijas nosacījumiem. Kļūdas tiek saglabatas veidlapas klasē. Tā lai atkārtoti izvadot veidlabu skatā, tās varētu izvadīt.

```
public function addAction() {
    // ...
    // Izveido veidlapas objektu
    $form = new TaskForm();
    $request = $this->getRequest();

    if ($request->isPost()) {
        // ...
        // Izveido uzdevuma objektu lai padot validācijas nosacījumus vaidlapas objektam.
        $task = new Task();
        // Padod validācijas nosacījumus veidlapas objektam
        $form->setInputFilter($task->getInputFilter());
        // Uzstāda veidlapai saļemtos datus
        $form->setData($request->getPost());

        if ($form->isValid()) {
            // ...
        }
    }
    // ...
}
```

Ilustrācija 2.37 Veidlapas apstrādes piemērs.

Veidlapas pilnās struktūras izvadīšana notiek pilnībā PHP vidē.

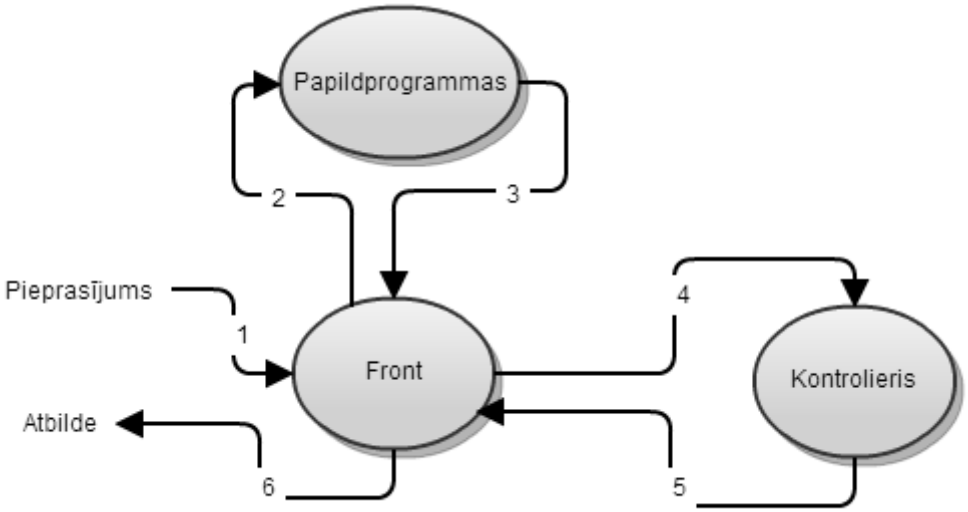
```
$form->setAttribute('action', $this->url('login'));
$form->prepare();

echo $this->form()->openTag($form);
echo $this->formRow($form->get('username'));
echo $this->formRow($form->get('password'));
echo $this->formSubmit($form->get('submit'));
echo $this->form()->closeTag();
```

Ilustrācija 2.38 Veidlapas izvadīšana.

2.4.6. Kodola plūsmas

„index.php" pieņem pieprasījumus un padod tos kodola klasei. Kodola klase „Front" pieņiem pieprasījumus uz sūta atbildes. Saņemot pieprasījumus tā palaiž daudzas papildprogrammas, pieslēdz maršrutēšanu un padod pieprasījumu attiecīgajam kontrolierim. Kontrolieris pēc apstrādes atgriež sagatavoto atbildi kodola klasei, kura sūta to atpakaļ pieprasījuma veicējam.



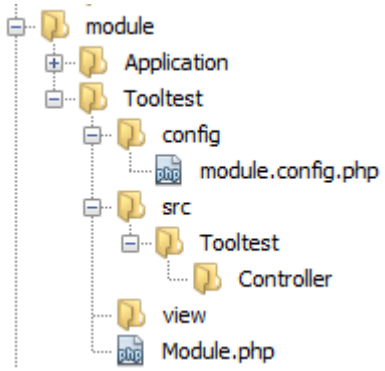
Ilustrācija 2.39 Stipri saspiests, datu plūsmu secīmas modelis.

2.4.7. Komponentes

Zend piedāvā vairākas komponentes, viena no tām ir Doctrine2 komponente. Publiski ir pieejamas daudz neoficiālās komponentes. Kā arī TWIG programmēšanas valodu var pieinstalēt kā komponenti.

2.4.8. ZF2 rīki

Ir pieejama rīku paka priekš ZF2. Pakete piedāvā vairākas konsoles komandas. Komandas izpildās PHP vidē.

Komanda	Apraksts
modules	Parāda visus pieslēgtos moduļus. Pētījumā vienmēr tika izvadīts, ka nav neviena moduļa, kaut arī projektā ir moduļi.
version	Izvada ZF2 versiju.
diag	Izvada norādītā moduļa diagnostiku.
create project <path>	Izveido jaunu ZF2 projektu.
create module <name>	Izveido jaunu moduli, kurš tiek pieslēgts projekta autolouderim, bet realizāciju tukšām datnēm. 
classmap generate <directory> <classmap_file>	Ģenerē klašu sarakstu PHP masīvā.

Mēģinot ģenerēt klašu karti, saskāros ar kļūdu. Ceļi līdz datnēm ir nepareizi. Iespējams tādēļ, ka nav izmantots Zend serveris, vai tādēļ, ka izstrāde tika veikta Windows operētājsistēmas vidē.

```
<?php
// Generated by Zend Framework 2
return array(
    'Application\Controller\CategoryController' => __DIR__ .
'/../../../../../C:\wamp\www\beinarovic.lv\zend\module\Application\src\Appl
ication\Controller/C:/wamp/www/beinarovic.lv/zend/module/Application/src/Ap
plication/Controller/CategoryController.php',
    'Application\Controller\IndexController'    => __DIR__ .
'/../../../../../C:\wamp\www\beinarovic.lv\zend\module\Application\src\Appl
```

```
ication\Controller/C:/wamp/www/beinarovic.lv/zend/module/Application/src/Ap
plication/Controller/IndexController.php',
    'Application\Controller\TaskController'     => __DIR__ .
'/../../../../../C:\wamp\www\beinarovic.lv\zend\module\Application\src\Appl
ication\Controller/C:/wamp/www/beinarovic.lv/zend/module/Application/src/Ap
plication/Controller/TaskController.php',
);
```

3. SALĪDZINĀJUMS

Galvenie salīdzināšanas kritēriji ir:

1. Jābūt labai un pieejama dokumentācijai.
2. Ietvaram jābūt viegli saprotamam, tā lai jaunie letotāji ātri varētu sākt izstrādāt ar tā palīdzību.
3. Ietvaram jābūt vienkāršiem risinājumiem, kuri varētu palīdzēt risināt klasiskās problēmas, tīmekļa izstrādē. Ārti var izveidot nelielu tīmekļa vietni.
4. Ietvaram jābūt piemērotam lielu projektu izveidošanai un uzturēšanai.

3.1. Dokumentācija

ZF2 oficiālā mājaslapā ir daudz dokumentācijas, bet viņa ir domāta tiem, kas jau ir strādājuši ar šo ietvaru, vai nu ar vecākām versijām kā Zend Framework 1. Tā ir balstīta uz konkrēta piemēra, bet paskaidrojumos uzreiz tiek stāstīts kā tas viss ir apstrādāts dziļi ietvara klasēs. Nav norādīts kādās datnēs ievieto piemēra kodu, nezinot Zend pierasto struktūru, ir grūti to saprast. Tas nav īpaši saprotami, jo sākotnēji mēģinot iemācīties ietvau svarīgi vienkārši saprast kā ar to var strādāt.

Toties Symfony2 mājaslapā ir tā sauktā „The Book”, dokumentācija, kurā ir aprakstīts kā strādāt ar šo ietvaru. Vienkārši piemēri un skaidri aprakstīts kas ko dara. Iedziļinoties var ar nākamām, tā sauktām „grāmatām”. Tiek piedāvāta „The Cookbook”, kur paskaidro kā paveikt grūtākus uzdevumus un iedziļinās sīkumos. „The Components” ir dokumentācija par vairākām piedāvātām komponentēm. Kā arī viņu mājaslapā ir vairākas dokumentāciju pakas.

Ilustrācija 3.1 Symfony2 mājaslapas dokumentāciju paku saraksts.

3.2. Lietošanas vienkāršība

ZF2 pēc instalācijas, ir gatavs darbam. Mainīgie uzstādījumi tiek pieprasīti tikai izmantošanas gadījumā. Tomēr Symfony2 katrā pieprasījumā pārbauda vai visi parametri ir pieejami. Pie sākotnējās palaišanas, Symfony2 piedāvā ievadīt visus nepieciešamos parametrus, tīmekļa vidē. ZF2 var trāpīties izņēmuma ekrāns par neievadītiem parametriem, toties Symfony2 pabrīdinās par parametru uzstādīšanu pirmajā palaišanas reizē. Daudz patīkamāk ir ievadīt visus parametrus tīmekļa vidē. Kā arī Symfony2 tiek izmantots YAML uzstādījumiem, kas ir daudz saskatamāks par PHP masīvu un aizņem daudz mazāk vietas.

```
return array(
    'modules' => array(
        'Application',
    ),
    'module_listener_options' => array(
        'module_paths' => array(
            './module',
            './vendor',
        ),
        'config_glob_paths' => array(
            'config/autoload/{,*.}{global,local}.php',
        ),
    ),
);
```

```
framework:
    secret:          %secret%
    router:
        resource: "%kernel.root_dir%/config/routing.yml
        strict_requirements: %kernel.debug%
    form:            ~
    csrf_protection: ~
    validation:      { enable_annotations: true }
    templating:
        engines: ['twig']
    default_locale:  "%locale%"
    trusted_proxies: ~
    session:         ~
    fragments:       ~
```

Ilustrācija 3.2 Uzstādījumu salīdzinājums. No labās puses Symfony2 uzstādījumi, bet no kreisās ZF2 uzstādījumi.

Datu bāzes pieprasījumi ZF2 ietvarā notiek ļoti vienkārši, kā arī Symfony2 ietvarā. Toties Doctrine2 un Symfony2 piedāvātā konsole dod priekšroku Symfony2 ietvaram. Kopā ar Symfony2 konsoli, pat sarežģītāko datu bāzi, izveidot vai pieslēgt, var ļoti ātri. Visu var

automātiski ģenerēt. Doctrine2 ir ļoti sarežģīta sistēma, sākotnēji tā var būt nepierasta, bet tai ir ļoti laba dokumentācija kura skaidri māca kā ar to strādāt. Toties ZF2 saskarsme ar datu bāzi ir krietni skaidrāka. Patstāvīgi jāraksta metodes datu uzstādīšanai, sarežģītākiem pieprasījumiem jābūt SQL formātā. Toties Symfony2 izmanto DQL, kas krietni atvieglo pieprasījumu veidošanu. ZF2 piedāvā pieslēgt Doctrine2 kā komponenti, bet tā nevar strādāt tik labi kā Symfony2 ietvarā, jo Symfony2 ir balstās uz Doctrine2.

Darbs ar veidlapām abiem ietvariem ir ļoti ērts. ZF2 viss notiek nedaudz skaidrāk nekā Symfony2, jo Symfony2 izmanto papildus objektus lai to visu realizēt, un nav skaidrs kā kas notiek. Toties ZF2 datu plūsma veidlapās ir acīmredzama.

Skatu veidošanā TWIG dod ārkārtīgi lielu priekšroku Symfony2 ietvaram. TWIG programmēšanas valoda ir ārkārtīgi vienkārša un metodes atgādina PHP vidi. Tā aizņem daudz mazāk vietas, attiecīgi mazāk laika aizņem rakstīšana tajā. ZF2 piedāvā ārkārtīgi daudz funkciju skatiem, dod iespēju automātiski ģenerēt struktūras elementus un sakārtošanas sistēma novērš koda dublēšanos. Abi ietvari piedāvā labu risinājumu skatu veidošanai, tomēr priekšroka ir TWIG valodai, jo tā ir maksimāli vienkāršota, domāta tieši skatu veidošanai.

Pieslēdzot assetus Symfony2 ietvarā, izstrādes procesā, aiziet daudz laika uz tā ģenerēšanu, toties tie tiek izmantoti labākā formā. ZF2 piedāvā funkcijas, assetu pieslēdzošo elementu ģenerēšanai, un asseti tiek glabāti pierastā tīmekļa izstrādes veidā. Task krietni atvieglina izstrādes procesu tiem, kas nav pieraduši pie Symfony2 assetu sistēmas.

Sākotnēji Symfony2 ir pieslēgtas daudzas komponentes, lai varētu uzreiz sākt izstrādāt, nedomājot par atsevišķu daļu pieslēgšanu. Priekš katra tīmekļa pieprasījuma, Symfony2 veic ļoti daudz darbību. Dēļ pareizas kešošanas tas nav pamanāms, bet veidojot sīku projektu notiek pārāk daudz lieka. Tas pataisa Symfony2 par neatbilstošu ietvaru nelielām tīmekļa vietnēm. ZF2 kodolā arī notiek ļoti daudz darbību, kā arī kešošana to kompensē.

Abiem ietvariem tiek piedāvātas ļoti daudzas komponentes, tas var stipri paātrināt kādu grūtāku izstrādes procesu, atrodot jau gatavu risinājumu.

Symfony2 pieraksta visas datu plūsmas un dod iespēju detalizēti sekot tai tīmekļa interfeisā. Tas nostrādā kā milzīgs pluss, jo nosekot datu plūsmām lielā sistēmā nav vienkārši.

Assetic sistēma, ar Yui filtriem, Symfony2 ietvarā stipri uzlabo projekta kvalitāti, kamēr ZF2 izmanto standarta variantu.

Abiem ietvariem ir iebūvētā kešošana, tas ļaus sarežģītiem pieprasījumiem izpildīties krietni ātrāk.

SECINĀJUMI

ZF2 ietvara oficiālā dokumentācijā ir ļoti sarešģīta, Symfony2 piedāvā krietni vienkāršāku un ar labākiem piemēriem, dokumentāciju. No tā seko ka vienkāršāk ir sākt strādāt ar Symfony2 ietvaru, nekā ar ZF2.

Abi ietvari ir ļoti spēcīgi, nav jēgas izmantot kādu no viņiem nelielas tīmekļa vietnes izstrādei. Šie ietvari ir domāti sarežģīto, vai nu neordināro tīmekļa vietņu veidošanai. Tā kā ZF2 sākotnējais apjoms, bez papildus komponentēm, ir nedaudz mazāks, nelielām tīmekļa vietnēm tas var būt nedaudz vairāk piemērots.

Toties vieglāka izstrāde var būt Symfony2 ietvarā, balstoties uz tādām komponentēm kā TWIG un plašu izvēli koda generēšanai. Symfony2 ir sakomplektēts no visvairāk pieprasītākajām risinājumu metodēm, tas to padara par ļoti efektīvu.

ZF2 izstrādātāji pieturējās pie pilnīgas PHP valodas lietošanas. Tas nostrādā kā pluss, tiem kas nav pārāk pieredzējuši ar programmēšanas valodām, jo nav vajadzības pielietot vairākas valodas.

IZMANTOTO AVOTU SARAKSTS

1. Why Our Business Choose Symfony2 Over Any Other PHP Framework by Alessandro Nadalin [tiešsaite] [skatīts 27.05.2013]. Piejams: http://odino.org/why-we-choose-symfony2-over-any-other-php-framework/
2. Top 5 PHP Frameworks Infographic by Helen Romanenko [tiešsaite] [skatīts 15.04.2013]. Piejams: http://php.dzone.com/articles/top-5-php-frameworks
3. PHP Symfony2 Framework - plugin detail [tiešsaite] [skatīts 2.04.2013]. Pieejams: http://plugins.netbeans.org/plugin/40565/php-symfony2-framework
4. Jaunu lietotāju atsauksmes par ZF2 [tiešsaite] [skatīts 6.04.2013]. Pieejamas: http://stackoverflow.com/questions/10182000/zend-framework-2-for-a-zend-framework-newbie
5. About Yii [tiešsaite] [skatīts 6.04.2013]. Pieejams: http://www.yiiframework.com/about/
6. Symfony2 vs Zend Framework 2 through the eyes of a novice by Idipous moderators [tiešsaite] [skatīts 15.04.2013]. Piejams: http://www.idipous.net/symfony2-vs-zend-framework-2-through-the-eyes-of-a-novice/
7. Symfony2 and HTTP Fundamentals [tiešsaite] [skatīts 10.05.2013]. Pieejams: http://symfony.com/doc/2.2/book/http_fundamentals.html
8. How to generate Entities from an Existing Database [tiešsaite] [skatīts 9.05.2013]. Pieejams: http://symfony.com/doc/current/cookbook/doctrine/reverse_engineering.html
9. How to create a custom User Provider [tiešsaite] [skatīts 10.05.2013]. Pieejams: http://symfony.com/doc/current/cookbook/security/custom_provider.html
10. How to customize your Form Login [tiešsaite] [skatīts 10.05.2013]. Pieejams: http://symfony.com/doc/current/cookbook/security/form_login.html
11. Validation [tiešsaite] [skatīts 23.05.2013]. Pieejams: http://symfony.com/doc/current/book/validation.html
12. @Route and @Method [tiešsaite] [skatīts 23.05.2013]. Pieejams: http://symfony.com/doc/current/bundles/SensioFrameworkExtraBundle/annotations/routing.html
13. How to Embed a Collection of Forms [tiešsaite] [skatīts 23.05.2013]. Pieejams: http://symfony.com/doc/2.2/cookbook/form/form_collections.html

14. Symfony2 dokumentāciju saraksts [tiešsaite] [skatīts 30.05.2013]. Pieejams: http://symfony.com/doc/current/index.html
15. Zend Framework 2 instalation [tiešsaite] [skatīts 23.05.2013]. Pieejams: https://github.com/zendframework/ZendSkeletonApplication#using-composer-recommended
16. Get started with Zend Framework 2 [tiešsaite] [skatīts 23.05.2013]. Pieejams: http://framework.zend.com/manual/2.0/en/user-guide/skeleton-application.html
17. Routers and Controllers [tiešsaite] [skatīts 24.05.2013]. Pieejams: http://framework.zend.com/manual/2.0/en/user-guide/routing-and-controllers.html
18. Routing [tiešsaite] [skatīts 30.05.2013].Pieejams: http://framework.zend.com/manual/2.0/en/modules/zend.mvc.routing.html
19. Database and models [tiešsaite] [skatīts 27.05.2013]. Pieejams: http://framework.zend.com/manual/2.0/en/user-guide/database-and-models.html
20. Forms and actions [tiešsaite] [skatīts 27.05.2013]. Pieejams: http://framework.zend.com/manual/2.0/en/user-guide/forms-and-actions.html
21. Database Table Authentication [tiešsaite] [skatīts 26.05.2013]. Pieejams: http://framework.zend.com/manual/2.1/en/modules/zend.authentication.adapter.dbtable.html
22. Zend Server for Developers [tiešsaite] [skatīts 31.05.2013]. Pieejams: http://www.zend.com/en/products/server/development
23. Zend Framework 2 Tool [tiešsaite] [skatīts 31.05.2013]. Pieejams: https://github.com/zendframework/ZFTool
24. Zend_Db_Select [tiešsaite] [skatīts 02.06.2013]. Pieejams: http://framework.zend.com/manual/1.12/en/zend.db.select.html
25. Working with Associations [tiešsaite] [skatīts 23.05.2013]. Pieejams: http://docs.doctrine-project.org/en/2.0.x/reference/working-with-associations.html
26. ZfcTwig Module for Zend Framework 2 [tiešsaite] [skatīts 02.06.2013]. Pieejams: https://github.com/ZF-Commons/ZfcTwig
27. Gliffy, diagrammu veidošanas rīks. Pieejams: https://www.gliffy.com/gliffy/
28. GitHub, versiju kontrole. Pieejams: https://github.com/
29. NetBeans, izstrādes rīks. Pieejams: https://netbeans.org/downloads/

30. Terminoloģijas portāls. Pieejams: http://www.termnet.lv/

31. Twitter Bootstrap. Pieejams: http://twitter.github.io/bootstrap/

32. JQuery datetime extension [tiešsaite] [skatīts 23.05.2013]. Pieejams: http://trentrichardson.com/examples/timepicker/

PIELIKUMI

1. PIELIKUMS TĪMEKĻA VIETŅU UN KODA PIEEJAMĪBA

Tīmekļa vietnes var atrast pēc sekojošām adresēm:

- Symfony2 ietvarā izveidotā tīmekļa vietne – http://symfony.beinarovics.lv
- Zend Framework 2 ietvarā izveidotā vietne – http://zend.beinarovics.lv

Ir pieejami publiskie GitHub repozitoriji, kuros glabājas kods. Tos var apskatīt pēc sekojošām adresēm:

- Symfony2 ietvarā izveidotās tīmekļa vietnes repozitorijs – https://github.com/ebeinarovics/todo_symfony
- Zend Framework 2 ietvarā izveidotās tīmekļa vietnes repozitorijs – https://github.com/ebeinarovics/todo_zend

2. PIELIKUMS IZVEIDOTĀS TĪMEKĻA VIETNES KOD SYMFONY2 IETVARĀ

Config.yml

```
imports:
    - { resource: parameters.yml }
    - { resource: security.yml }

# ietvara uzstādījumi
framework:
    secret:          %secret%
    router:
        resource: "%kernel.root_dir%/config/routing.yml"
        strict_requirements: %kernel.debug%
    form:            ~
    csrf_protection: ~
    validation:      { enable_annotations: true }
    templating:
        engines: ['twig']
    default_locale:  "%locale%"
    trusted_proxies: ~
    session:         ~
    fragments:       ~

# Twig uzstādījumi
twig:
    debug:            %kernel.debug%
    strict_variables: %kernel.debug%

# Assetic Configuration
assetic:
    debug:          %kernel.debug%
    use_controller: false
    bundles:        [TodoClientBundle]
    filters:
        cssrewrite: ~

# Doktrīnes uzstādījumi
doctrine:
    dbal:
        driver:   %database_driver%
        host:     %database_host%
        port:     %database_port%
        dbname:   %database_name%
        user:     %database_user%
        password: %database_password%
        charset:  UTF8
    orm:
        auto_generate_proxy_classes: %kernel.debug%
        auto_mapping: true

# Swiftmailer uzstādījumi
swiftmailer:
    transport: %mailer_transport%
    host:      %mailer_host%
    username:  %mailer_user%
    password:  %mailer_password%
    spool:     { type: memory }
```

```
# Servisu reģistrēšana
services:
    todo.manager.user:
        class: Todo\Bundle\CoreBundle\Manager\UserManager
        arguments: []
    twig.extension.text:
        class: Twig_Extensions_Extension_Text
        tags:
            - { name: twig.extension }

# Injekciju uzstādījumi
jms_di_extra:
    locations:
        all_bundles: false
        bundles: [TodoCoreBundle]
        directories: ["%kernel.root_dir%/../src"]
```

parameters.yml

```
parameters:
    database_driver: pdo_mysql
    database_host: 127.0.0.1
    database_port: null
    database_name: todo
    database_user: root
    database_password: messager
    mailer_transport: smtp
    mailer_host: 127.0.0.1
    mailer_user: null
    mailer_password: null
    locale: en
    secret: 7d29b77d7052f8d1b7cb88851bb7bd2f3a1a4c5d
    database_path: null
```

routing.yml

```
# maršruti autorizācijai un reģistrācijai
todo_security:
    resource: "@TodoSecurityBundle/Controller/"
    type:     annotation
    prefix:   /
# Maršruti klientu daļai
todo_client:
    resource: "@TodoClientBundle/Controller/"
    type:     annotation
    prefix:   /

logout:
    pattern:   /logout
```

security.yml

```
security:
    encoders:
        Todo\Bundle\CoreBundle\Entity\User:
            algorithm:        sha1
            encode_as_base64: false
            iterations:       1
    # lietotāju meklēšanai
    providers:
```

```
        database:
            entity: { class: TodoCoreBundle:User, property: username }

        in_memory:
            memory:
                users:
                    admin: { password: qwerty, roles: [ 'ROLE_ADMIN' ] }

    firewalls:
        login:
            pattern:    ^/
            form_login:
                check_path: todo_login_check
                login_path: todo_login
                always_use_default_target_path: true
                default_target_path: todo_list
                provider: database
                target_path_parameter: todo_list
            logout:
                path:   /logout
                target: /
            anonymous: ~

# Atļautās zonas


    access_control:
        - { path: ^/todo/*, roles: ROLE_USER }
        - { path: ^/, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/sugnup, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/login-check, roles: IS_AUTHENTICATED_ANONYMOUSLY }
```

CategoryController.php

```php
<?php

namespace Todo\Bundle\ClientBundle\Controller;

use Todo\Bundle\ClientBundle\Controller\CoreController;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;
use Todo\Bundle\CoreBundle\Form\CategoryType;
use Todo\Bundle\CoreBundle\Entity\Category;

/**
 * @Route("/todo/category")
 */
class CategoryController extends CoreController
{
    /**
     * @Route("/list", name="todo_category_list")
     * @Template()
     */
    public function listAction()
    {
        $repo = $this->getCategoryRepository();

        $categories = $repo->findBy(array(
            'user'  => $this->getUser()
        ));

        return array(
            'categories'    => $categories
```

```
        );
    }

    /**
     * @Route("/create", name="todo_category_create")
     * @Template()
     */
    public function createAction()
    {
        $category = new Category();
        $form = $this->createForm(new CategoryType(), $category);
        $request = $this->getRequest();

        if($request->getMethod() == 'POST') {
            $form->bindRequest($request);
            if($form->isValid()) {
                $em = $this->getDoctrine()->getEntityManager();

                $category->setUser($this->getUser());

                $em->persist($category);
                $em->flush();

                return $this->redirect($this-
>generateUrl('todo_category_list'));
            }
        }

        return array(
            'form'   => $form->createView()
        );
    }

    /**
     * @Route("/remove/{id}", name="todo_category_remove")
     * @Template()
     */
    public function removeAction($id)
    {
        $category = $this->getCategoryRepository()->find($id);
        if ($category->getUser() == $category->getUser()) {
            $em = $this->getDoctrine()->getManager();

            $em->remove($category);
            $em->flush();
        }

        return $this->redirect($this->generateUrl('todo_category_list'));
    }
}
```

CoreController.php

```
<?php

namespace Todo\Bundle\ClientBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Todo\Bundle\CoreBundle\Repository\UserRepository;
use Todo\Bundle\CoreBundle\Manager\UserManager;
use Todo\Bundle\CoreBundle\Repository\TaskRepository;
use Doctrine\ORM\EntityRepository;
```

```
class CoreController extends Controller
{
    /**
     * @return UserRepository
     */
    public function getUserRepository()
    {
        return $this->getDoctrine()
                    ->getRepository('TodoCoreBundle:User');
    }

    /**
     * @return TaskRepository
     */
    public function getTaskRepository()
    {
        return $this->getDoctrine()
                    ->getRepository('TodoCoreBundle:Task');
    }

    /**
     * @return EntityRepository
     */
    public function getCategoryRepository()
    {
        return $this->getDoctrine()
                    ->getRepository('TodoCoreBundle:Category');
    }

    /**
     * @return UserManager
     */
    public function getUserManager()
    {
        return $this->get('todo.manager.user');
    }
}
```

HomepageController.php

```
<?php

namespace Todo\Bundle\ClientBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;
use Todo\Bundle\CoreBundle\Form\UserType;

class HomepageController extends Controller
{
    /**
     * @Route("/", name="todo_homepage")
     * @Template()
     */
    public function indexAction()
    {
        $registrationForm = $this->createForm(new UserType());

        return array(
            'registration'  => $registrationForm->createView()
        );
    }
```

```
}
```

IncludeController.php

```
<?php

namespace Todo\Bundle\ClientBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;

class IncludeController extends Controller
{
    /**
     * @Template()
     */
    public function headerAction()
    {
        return array();
    }
}
```

ListController.php

```
<?php

namespace Todo\Bundle\ClientBundle\Controller;

use Todo\Bundle\ClientBundle\Controller\CoreController;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;
use Todo\Bundle\CoreBundle\Form\TaskType;
use Todo\Bundle\CoreBundle\Entity\Task;
use Todo\Bundle\CoreBundle\Entity\Category;
use Todo\Bundle\CoreBundle\Form\CommentType;
use Todo\Bundle\CoreBundle\Entity\Comment;
use Todo\Bundle\CoreBundle\Entity\CustomField;

/**
 * @Route("/todo/task")
 */
class ListController extends CoreController
{
    /**
     * @Route("/", name="todo_list")
     * @Template()
     */
    public function indexAction()
    {
        $doctrine = $this->getDoctrine();
        $categoryRepo = $doctrine-
>getRepository('TodoCoreBundle:Category');
        /**
         * Nolasa no datu bāzes katgorijas
         */
        $categories = $categoryRepo->findBy(array(
            'user'  => $this->getUser()
        ));

        /**
         * Padod tās skatam
```

```
         */
        return array(
            'categories'    => $categories
        );
    }

    /**
     * @Template()
     */
    public function tasksForCategoryAction(Category $category)
    {
        $doctrine = $this->getDoctrine();

        $categoryRepo = $doctrine->getRepository('TodoCoreBundle:Task');
        /**
         * Nolasa visus uzdevumus priekš kategorijas
         */
        $tasks = $categoryRepo->getByUserAndCategory($this->getUser(),
$category);

        return array(
            'tasks'    => $tasks
        );
    }

    /**
     * @Route("/create", name="todo_task_create")
     * @Template()
     */
    public function createAction()
    {
        $task = new Task(); // izveido jaunu uzdevumu un veidlapu priekš tā
        $form = $this->createForm(new TaskType($this->getUser()), $task);
        $request = $this->getRequest();

        if ($request->getMethod() == 'POST') {
            // ja pieprasījuma veidz ir POST, apstrādā to
            $form->bindRequest($request);
            if ($form->isValid()) {
                $em = $this->getDoctrine()->getEntityManager();

                $task->setUser($this->getUser());

                $em->persist($task);

                foreach ($task->getCustomField() as $custom) {
                    $custom->setTask($task);
                    $em->persist($custom);
                }
                /// saglabā datu bāzē
                $em->flush();

                return $this->redirect($this->generateUrl('todo_task_show',
array(
                    'id'    => $task->getId()
                )));
            }
        }

        return array(
            'form'  => $form->createView()
        );
    }
```

```
    /**
     * @Route("/remove/{id}", name="todo_task_remove")
     * @Template()
     */
    public function removeAction($id)
    {
        $task = $this->getTaskRepository()->find($id);
        if ($task->getUser() == $this->getUser()) {
            $em = $this->getDoctrine()->getManager();
            // Izdzēš ierakstus
            $em->remove($task);
            $em->flush();
        }

        return $this->redirect($this->generateUrl('todo_list'));
    }

    /**
     * @Route("/edit/{id}", name="todo_task_edit")
     * @Template()
     */
    public function editAction($id)
    {
        $task = $this->getTaskRepository()->find($id);
        $form = $this->createForm(new TaskType($this->getUser()), $task);
        $request = $this->getRequest();

        $descr = $task->getDescription();
        $categ = $task->getCategory();
        $evtim = $task->getEventTime();
        $progr = $task->getProgress();
        $statu = $task->getStatus();
        $title = $task->getTitle();

        if($request->getMethod() == 'POST') {
            $form->bindRequest($request);
            if($form->isValid()) {
                $em = $this->getDoctrine()->getManager();

                $task->setUser($this->getUser());

                $em->persist($task);

                foreach($task->getCustomField() as $custom) {
                    $custom->setTask($task);
                    $em->persist($custom);
                }

                $changes = '';

                if($title !== $task->getTitle()) {
                    $changes .= '<br> Title: '.$title;
                }
                if($descr !== $task->getDescription()) {
                    $changes .= '<br> Description: '.$descr;
                }
                if($evtim !== $task->getEventTime()) {
                    if($evtim) {
                        $et = $evtim->format('d-m-Y @ H:i');
                    } else {
                        $et = null;
                    }
                    $changes .= '<br> Event time: '.$et;
                }
```

```
                if($categ !== $task->getCategory()) {
                    $changes .= '<br> Category: '.$categ->getName();
                }
                if($statu !== $task->getStatus()) {
                    $changes .= '<br> Status: '.$statu->getName();
                }
                if($progr !== $task->getProgress()) {
                    $changes .= '<br> Progress: '.$progr;
                }

                if ($changes !== '') {
                    $comment = new Comment();
                    $comment->setTask($task);
                    $comment->setChanged($changes);
                    $comment->setCreatedAt(new \DateTime());
                    $em->persist($comment);
                }
                $em->flush();

                return $this->redirect($this->generateUrl('todo_task_show',
array(
                    'id'    => $task->getId()
                )));
            }
        }

        return array(
            'form'   => $form->createView(),
            'id'     => $id
        );
    }

    /**
     * @Route("/show/{id}", name="todo_task_show")
     * @Template()
     */
    public function showAction($id)
    {
        $request = $this->getRequest();
        $task = $this->getTaskRepository()->find($id);
        $comment = new Comment();

        $form = $this->createForm(new CommentType(), $comment);

        if($request->getMethod() == 'POST') {
            $form->bindRequest($request);
            if($form->isValid()) {
                $comment->setTask($task);
                $comment->setCreatedAt(new \DateTime());

                $em = $this->getDoctrine()->getManager();
                $em->persist($comment);
                $em->flush();

                $form = $this->createForm(new CommentType(), new
Comment());
            }
        }

        return array(
            'task'   => $task,
            'form'   => $form->createView()
        );
    }
```

```
}
```

Category/create.html.twig

```
{% extends 'TodoClientBundle::layout.html.twig' %}

{% block content %}
    <div class="row-fluid">
        <div class="span12">
            <form action="{{ path('todo_category_create') }}" method="post"
{{ form_enctype(form) }}>
                {{ form_widget(form) }}
                <input type="submit" class="btn btn-primary">
            </form>
        </div>
    </div>
{% endblock content %}
```

Category/list.html.twig

```
{% extends 'TodoClientBundle::layout.html.twig' %}

{% block content %}
    <p class="text-center">
        <a href="{{ path('todo_category_create') }}" class="btn btn-
success">Create</a>
    </p>
    <div class="row-fluid">
        <div class="span12">
            <table class="table table-striped">
                {% for category in categories %}
                <tr>
                    <td>{{ category.name }}</td>
                    <td>
                        <a class="btn btn-small"
                           href="{{ path('todo_category_remove', { 'id':
category.id }) }}"
                           >Remove</a>
                    </td>
                </tr>
                {% endfor %}
            </table>
        </div>
    </div>
{% endblock content %}
```

Homepage/index.html

```
{% extends 'TodoClientBundle::layout.html.twig' %}

{% if last_username is not defined %}
    {% set last_username = '' %}
{% endif %}

{% block content %}
    <div class="row-fluid">
        <div class="span12">
            {% if error.message is defined %}
                <div class="alert alert-error">
                    <p class="text-center">{{ error.message }}</p>
```

```
                </div>
            {% elseif error is defined %}
                <div class="alert alert-error">
                    <p class="text-center">{{ error }}</p>
                </div>
            {% elseif success is defined %}
                <div class="alert alert-success">
                    <p class="text-center">You have successfully
registered</p>
                </div>
            {% endif %}
            <div class="row-fluid">
                <div class="span6 well">
                    <h3>Login</h3>
                    <form action="{{ path('todo_login_check') }}"
method="post">
                        <label for="username">Username:</label>
                        <input type="text" id="username" name="_username"
value="{{ last_username }}" />

                        <label for="password">Password:</label>
                        <input type="password" id="password"
name="_password" />

                        <input type="hidden" name="_target_path"
value="account" />
                        <br/>

                        <input class="btn btn-info" type="submit"
name="login" />
                    </form>
                </div>
                <div class="span6 well">
                    <h3>Registration</h3>
                    <form action="{{ path('todo_signup') }}" method="post"
{{ form_enctype(registration) }} autocomplete="off">
                        {{ form_widget(registration) }}
                        <input type="submit" class="btn btn-primary"
value="register">
                    </form>
                </div>
            </div>
        </div>
    </div>
{% endblock content %}
```

Include/header.html.twig

```
<div class="container">
    <div class="navbar-inner">
        <a class="brand" href="#">Todo</a>
        <ul class="nav nav-pills">
            {% block header %}
                {% if app.user %}
                    <li><a href="{{ path('todo_list') }}">Home</a></li>
                    <li><a href="{{ path('todo_category_list')
}}">Categories</a></li>
                    <li class="span3 "><a href="{{ path('todo_task_create')
}}">Crete task</a></li>
                    <li class="active">
                        <a href="#">
                            {{ app.user.firstname }}
                            {{ app.user.lastname }}
```

```
                            ({{ app.user.username }})
                        </a>
                    </li>
                    <li><a href="{{ path('logout') }}">Logout</a></li>
                {% else %}
                    <li class="active"><a href="{{ path('todo_homepage')
}}">Home</a></li>
                {% endif %}
            {% endblock header %}
        </ul>
    </div>
</div>
```

List/create.html.twig

```
{% extends 'TodoClientBundle::layout.html.twig' %}

{% block content %}
    <div class="row-fluid">
        <h2>Creating task</h2>
        <div class="span12">
            <form action="{{ path('todo_task_create') }}" method="post" {{
form_enctype(form) }}>
                <div class="row">
                    <div class="span4">
                        <h3>Task</h3>
                        {{ form_row(form.title) }}
                        {{ form_row(form.description) }}
                        {{ form_row(form.status) }}
                        {{ form_row(form.progress) }}
                        {{ form_row(form.eventTime) }}
                        {{ form_row(form.category) }}
                    </div>
                    <div class="span8">
                        <h3>Custom fields</h3>
                        <ul class="tags" data-prototype="{{
form_widget(form.customField.vars.prototype)|e }}">
                        </ul>
                    </div>
                </div>
                {{ form_rest(form) }}
                <input type="submit" class="btn btn-primary">
            </form>
        </div>
    </div>
{% endblock content %}

{% block postjavascripts %}
    <script>
        setInterval(function(){
            $('.new-custom-field').each(function (){
                var value = $(this).find('.value');
                var option =
$(this).find('select').find('option:selected');

                if($(option).text() == 'Date Time') {
                    $(value).datetimepicker({
                        dateFormat: 'dd-mm-yy',
                        separator: ' ',
                        minDate: new Date('-1Hours')
                    });
                } else {
                    $(value).datetimepicker("destroy");
```

```
                }
            });
        }, 500);

        $(function(){
            $(".datepicker").datetimepicker({
                dateFormat: 'dd-mm-yy',
                separator: ' ',
                minDate: new Date('-1Hours')
            });
        });

        // saraksts ar papildus laukiem
        var collectionHolder = $('ul.tags');

        // papildus lauka uzstādīšana
        var $addTagLink = $('<a href="#" class="add_tag_link">Add a custom
field</a>');
        var $newLinkLi = $('<li></li>').append($addTagLink);

        jQuery(document).ready(function() {
            // add the "add a tag" anchor and li to the tags ul
            collectionHolder.append($newLinkLi);

            // count the current form inputs we have (e.g. 2), use that as
the new
            // index when inserting a new item (e.g. 2)
            collectionHolder.data('index',
collectionHolder.find(':input').length);

            $addTagLink.on('click', function(e) {
                // prevent the link from creating a "#" on the URL
                e.preventDefault();

                // add a new tag form (see next code block)
                addTagForm(collectionHolder, $newLinkLi);
            });
        });

        function addTagForm(collectionHolder, $newLinkLi) {
            // Get the data-prototype explained earlier
            var prototype = collectionHolder.data('prototype');

            // get the new index
            var index = collectionHolder.data('index');

            // Replace '__name__' in the prototype's HTML to
            // instead be a number based on how many items we have
            var newForm = prototype.replace(/__name__/g, index);

            // increase the index with one for the next item
            collectionHolder.data('index', index + 1);

            // Display the form in the page in an li, before the "Add a
tag" link li
            var $newFormLi = $('<li class="new-custom-
field"></li>').append(newForm);
            $newLinkLi.before($newFormLi);
        }
    </script>
{% endblock postjavascripts %}
```

List/edit.html.twig

```
{% extends 'TodoClientBundle::layout.html.twig' %}

{% block content %}
    <div class="row-fluid">
        <h2>Updating task</h2>
        <div class="span12">
            <form action="{{ path('todo_task_edit', { 'id': id  }) }}"
method="post" {{ form_enctype(form) }}>
                <div class="row">
                    <div class="span4">
                        <h3>Task</h3>
                        {{ form_row(form.title) }}
                        {{ form_row(form.description) }}
                        {{ form_row(form.status) }}
                        {{ form_row(form.progress) }}
                        {{ form_row(form.eventTime) }}
                        {{ form_row(form.category) }}
                    </div>
                    <div class="span8">
                        <h3>Custom fields</h3>
                        <ul class="tags" data-prototype="{{
form_widget(form.customField.vars.prototype)|e }}">
                            {% for custom in form.customField %}
                                <li>{{ form_row(custom.key) }}</li>
                                <li>{{ form_row(custom.value) }}</li>
                                <li class="hide">{{
form_row(custom.fieldType) }}</li>
                            {% endfor %}
                        </ul>
                    </div>
                </div>
                {{ form_rest(form) }}
                <input type="submit" class="btn btn-primary">
            </form>
        </div>
    </div>
{% endblock content %}

{% block postjavascripts %}
    <script>
        setInterval(function(){
            $('.new-custom-field').each(function (){
                var value = $(this).find('.value');
                var option =
$(this).find('select').find('option:selected');

                if($(option).text() == 'Date Time') {
                    $(value).datetimepicker({
                        dateFormat: 'dd-mm-yy',
                        separator: ' ',
                        minDate: new Date('-1Hours')
                    });
                } else {
                    $(value).datetimepicker("destroy");
                }
            });
        }, 500);

        $(function(){
            $(".datepicker").datetimepicker({
                dateFormat: 'dd-mm-yy',
                separator: ' ',
                minDate: new Date('-1Hours')
            });
```

```
        });

        // Get the ul that holds the collection of tags
        var collectionHolder = $('ul.tags');

        // setup an "add a tag" link
        var $addTagLink = $('<a href="#" class="add_tag_link">Add a custom
field</a>');
        var $newLinkLi = $('<li></li>').append($addTagLink);

        jQuery(document).ready(function() {
            // add the "add a tag" anchor and li to the tags ul
            collectionHolder.append($newLinkLi);

            // count the current form inputs we have (e.g. 2), use that as
the new
            // index when inserting a new item (e.g. 2)
            collectionHolder.data('index',
collectionHolder.find(':input').length);

            $addTagLink.on('click', function(e) {
                // prevent the link from creating a "#" on the URL
                e.preventDefault();

                // add a new tag form (see next code block)
                addTagForm(collectionHolder, $newLinkLi);
            });
        });

        function addTagForm(collectionHolder, $newLinkLi) {
            // Get the data-prototype explained earlier
            var prototype = collectionHolder.data('prototype');

            // get the new index
            var index = collectionHolder.data('index');

            // Replace '__name__' in the prototype's HTML to
            // instead be a number based on how many items we have
            var newForm = prototype.replace(/__name__/g, index);

            // increase the index with one for the next item
            collectionHolder.data('index', index + 1);

            // Display the form in the page in an li, before the "Add a
tag" link li
            var $newFormLi = $('<li class="new-custom-
field"></li>').append(newForm);
            $newLinkLi.before($newFormLi);
        }
    </script>
{% endblock postjavascripts %}
```

List/index.html.twig

```
{% extends 'TodoClientBundle::layout.html.twig' %}

{% block content %}
    <p class="text-center">
        <a href="{{ path('todo_task_create') }}" class="btn btn-
success">New Task</a>
        <a href="{{ path('todo_category_create') }}" class="btn btn-
success">New Category</a>
    </p>
    <div class="row-fluid">
        <div class="span12">
            <h2>Dashboard</h2>
            {% for category in categories %}
                <h4>{{ category }}</h4>
                {% render
controller('TodoClientBundle:List:tasksForCategory', { 'category': category
}) %}
            {% endfor %}
        </div>
    </div>
{% endblock content %}
```

List/show.html.twig

```
{% extends 'TodoClientBundle::layout.html.twig' %}

{% block content %}
    <div class="row-fluid">
        <h1>
            <button class="btn btn-large disabled {% if task.status.active
%} btn-primary{% else %} btn-inverse{% endif %}">
                {{ task.status.name }}
            </button>
            {{ task.title }}
        </h1>
        <div class="span12">
            <p>
                {% if task.eventTime %}
                    <span class="btn disabled {% if task.status.active %}
btn-primary{% else %} btn-inverse{% endif %}">
                        {{ task.eventTime | date('d-m-Y @ H:i') }}
                    </span>
                {% endif %}
            </p>
            <div class="progress progress-striped{% if not
task.status.active %} progress-warning{% endif %}" title="{{ task.progress
}}%">
                <div class="bar" style="width: {{ task.progress ~ '%'
}};"></div>
            </div>

            <p{% if task.status.active %} class="text-info"{% endif %}>
                <a href="{{ path('todo_task_show', { 'id': task.id }) }}">
                    <strong>
                        {{ task.title }}
                    </strong>
                </a>
            </p>
            {% for custom in task.customField %}
```

```
            <p>
                <span class="span4"><strong>{{ custom.key
}}:</strong></span>
                <span class="span8">
                    {% if custom.fieldType.dataType == 'DATETIME' %}
                        <span class="badge{% if task.status.active %}
badge-info{% endif %}">
                            {{ custom.value }}
                        </span>
                    {% else %}
                        {{ custom.value }}
                    {% endif %}
                </span>
            </p>
            {% endfor %}

            <p>{{ task.description }}</p>

            <form action="{{ path('todo_task_show', { 'id': task.id }) }}"
method="post" {{ form_enctype(form) }}>
                {{ form_widget(form) }}
                <input type="submit" class="btn btn-primary"
value="Comment">
                <a href="{{ path('todo_task_edit', { 'id': task.id }) }}"
class="btn btn-small">Update</a>
                <a href="#" class="btn btn-small">Remove</a>
            </form>

            {% for comment in task.comments | reverse %}
                <div class="span12 well">
                    {% if comment.changed is null %}
                        <p>
                            <span class="badge{% if task.status.active %}
badge-info{% endif %}">
                                {{ comment.createdAt | date('d-m-Y @ H:i')
}}
                            </span>
                            <strong>Commented:</strong>
                            {{ comment.comment }}
                        </p>
                    {% else %}
                        <p>
                            <span class="badge{% if task.status.active %}
badge-info{% endif %}">
                                {{ comment.createdAt | date('d-m-Y @ H:i')
}}
                            </span>
                            <strong>Updated. Before update:</strong>
                            <br>
                            {{ comment.changed | raw }}
                        </p>
                    {% endif %}
                </div>
            {% endfor %}

        </div>
    </div>
{% endblock content %}
```

taskForCategory.html.twig

```
<table class="table">
    <thead>
```

```
        <tr>
            <td>time</td>
            <td>title</td>
            <td>status</td>
            <td>progress</td>
            <td></td>
        </tr>
    </thead>
    {% for task in tasks %}
        <tr {% if task.status.active %}class="info"{% endif %}>
            <td class="span2">
                <p>
                    {% if task.eventTime %}
                        <span class="badge{% if task.status.active %}
badge-info{% endif %}">
                            {{ task.eventTime | date('d-m-Y @ H:i') }}
                        </span>
                    {% endif %}
                </p>
            </td>
            <td class="span3">
                <p{% if task.status.active %} class="text-info"{% endif %}>
                    <a href="{{ path('todo_task_show', { 'id': task.id })
}}">
                        <strong>
                            {{ task.title | truncate(30) }}
                        </strong>
                    </a>
                </p>
            </td>
            <td class="span2">
                <p>
                    <span class="badge{% if task.status.active %} badge-
success{% endif %}">
                        {{ task.status.name }}
                    </span>
                </p>
            </td>
            <td class="span2">
                <div class="progress progress-striped{% if not
task.status.active %} progress-warning{% endif %}" title="{{ task.progress
}}%">
                    <div class="bar" style="width: {{ task.progress ~ '%'
}};"></div>
                </div>
            </td>
            <td class="span3">
                <a href="{{ path('todo_task_edit', { 'id': task.id }) }}"
class="btn btn-small">Update</a>
                <a href="{{ path('todo_task_remove', { 'id': task.id }) }}"
class="btn btn-small">Remove</a>
            </td>
        </tr>
        <tr>
            <td colspan="5" class="span12">
                <p{% if task.status.active %} class="text-info"{% endif
%}>{{ task.description | truncate(375) }}</p>
            </td>
        </tr>
    {% endfor %}
</table>
```

layout.html.twig

```
{% extends '::base.html.twig' %}

{% block stylesheets %}
    {{ parent() }}
    {% stylesheets
        "@TodoClientBundle/Resources/public/css/bootstrap.css"
        "@TodoClientBundle/Resources/public/css/jquery-ui-
1.10.3.custom.min.css"
    %}
        <link rel="stylesheet" href="{{ asset_url }}" />
    {% endstylesheets %}
{% endblock %}

{% block javascripts %}
    {{ parent() }}
    {% javascripts
        "@TodoClientBundle/Resources/public/js/jquery-1.9.1.js"
        "@TodoClientBundle/Resources/public/js/jquery-ui-
1.10.3.custom.min.js"
        "@TodoClientBundle/Resources/public/js/bootstrap.js"
        "@TodoClientBundle/Resources/public/js/time-picker-extension.js"
    %}
        <script src="{{ asset_url }}"></script>
    {% endjavascripts %}
{% endblock javascripts %}

{% block body %}
    <div class="navbar">
        {% include 'TodoClientBundle:Include:header.html.twig' %}
    </div>

    <div class="container">
        {% block content %}{% endblock content %}
        <div class="row-fluid">
            <div class="span9">
                <address>
                    <strong>(c) 2013 Rīga</strong><br>
                    Latvijas Universitāte<br>
                </address>
            </div>
            <div class="span3">
                <address>
                    <strong>Edmunds Beinarovičs</strong><br>
                    <a
href="mailto:edmund@beinarovic.lv">edmund@beinarovic.lv</a><br>
                    <abbr title="Phone">P:</abbr> +371 267 198 47
                </address>
            </div>
        </div>
    </div>
{% endblock body %}
```

AuthorCommand.php

```
<?php
namespace Todo\Bundle\CoreBundle\Command;

use Symfony\Bundle\FrameworkBundle\Command\ContainerAwareCommand;
use Symfony\Component\Console\Input\InputArgument;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;
use Symfony\Component\Console\Formatter\OutputFormatterStyle;
```

```
class AuthorCommand extends ContainerAwareCommand
{
    protected function configure()
    {
        $this
            ->setName('todo:project:author')
            ->setDescription('Shows the author of the project.')
            ->addArgument(
                'count',
                InputArgument::REQUIRED,
                'Count of times to display'
            )
        ;
    }

    protected function execute(InputInterface $input, OutputInterface
$output)
    {
        $this->loadStyles($output);
        $count = (int)$input->getArgument('count');

        if ($count < 1 || $count > 100) {
            $output->writeln("<error>Wrong count $count</error>");

            return;
        }

        while (($count--) > 0) {
            $output->writeln('The author is <success>Edmunds
Beinarovics</success>!');
        }
    }

    private function loadStyles(OutputInterface &$output) {
        $styles = array(
            'error' => new OutputFormatterStyle('red', null, array()),
            'success' => new OutputFormatterStyle('green', null,
array('bold'))
        );

        foreach($styles as $key => $style) {
            $output->getFormatter()->setStyle($key, $style);
        }
    }
}
```

AllowedUsers.php

```
<?php

namespace Todo\Bundle\CoreBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * AllowedUsers
 *
 * @ORM\Table(name="allowed_users")
 * @ORM\Entity
 */
class AllowedUsers
{
    /**
```

```
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="permissions", type="string", length=3,
nullable=false)
     */
    private $permissions;

    /**
     * @var string
     *
     * @ORM\Column(name="allowed_userscol", type="string", length=45,
nullable=true)
     */
    private $allowedUserscol;

    /**
     * @var \User
     *
     * @ORM\ManyToOne(targetEntity="User")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="user_id", referencedColumnName="id")
     * })
     */
    private $user;

    /**
     * @var \Task
     *
     * @ORM\ManyToOne(targetEntity="Task")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="task_id", referencedColumnName="id")
     * })
     */
    private $task;



    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set permissions
     *
     * @param string $permissions
     * @return AllowedUsers
     */
    public function setPermissions($permissions)
    {
```

```
        $this->permissions = $permissions;

        return $this;
    }

    /**
     * Get permissions
     *
     * @return string
     */
    public function getPermissions()
    {
        return $this->permissions;
    }

    /**
     * Set allowedUserscol
     *
     * @param string $allowedUserscol
     * @return AllowedUsers
     */
    public function setAllowedUserscol($allowedUserscol)
    {
        $this->allowedUserscol = $allowedUserscol;

        return $this;
    }

    /**
     * Get allowedUserscol
     *
     * @return string
     */
    public function getAllowedUserscol()
    {
        return $this->allowedUserscol;
    }

    /**
     * Set user
     *
     * @param \Todo\Bundle\CoreBundle\Entity\User $user
     * @return AllowedUsers
     */
    public function setUser(\Todo\Bundle\CoreBundle\Entity\User $user =
null)
    {
        $this->user = $user;

        return $this;
    }

    /**
     * Get user
     *
     * @return \Todo\Bundle\CoreBundle\Entity\User
     */
    public function getUser()
    {
        return $this->user;
    }

    /**
     * Set task
```

```
     *
     * @param \Todo\Bundle\CoreBundle\Entity\Task $task
     * @return AllowedUsers
     */
    public function setTask(\Todo\Bundle\CoreBundle\Entity\Task $task =
null)
    {
        $this->task = $task;

        return $this;
    }

    /**
     * Get task
     *
     * @return \Todo\Bundle\CoreBundle\Entity\Task
     */
    public function getTask()
    {
        return $this->task;
    }
}
```

Category.php

```
<?php

namespace Todo\Bundle\CoreBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Category
 *
 * @ORM\Table(name="category")
 * @ORM\Entity
 */
class Category
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=45, nullable=false)
     */
    private $name;

    /**
     * @var \User
     *
     * @ORM\ManyToOne(targetEntity="User")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="user_id", referencedColumnName="id")
     * })
     */
```

```
    private $user;

    /**
     * @ORM\OneToMany(mappedBy="category", targetEntity="Task",
cascade={"persist", "remove"})
     */
    private $task;

    public function __toString() {
        return $this->name;
    }

    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set name
     *
     * @param string $name
     * @return Category
     */
    public function setName($name)
    {
        $this->name = $name;

        return $this;
    }

    /**
     * Get name
     *
     * @return string
     */
    public function getName()
    {
        return $this->name;
    }

    /**
     * Set user
     *
     * @param \Todo\Bundle\CoreBundle\Entity\User $user
     * @return Category
     */
    public function setUser(\Todo\Bundle\CoreBundle\Entity\User $user =
null)
    {
        $this->user = $user;

        return $this;
    }

    /**
     * Get user
     *
     * @return \Todo\Bundle\CoreBundle\Entity\User
     */
```

```
    public function getUser()
    {
        return $this->user;
    }
    /**
     * Constructor
     */
    public function __construct()
    {
        $this->task = new \Doctrine\Common\Collections\ArrayCollection();
    }

    /**
     * Add task
     *
     * @param \Todo\Bundle\CoreBundle\Entity\Task $task
     * @return Category
     */
    public function addTask(\Todo\Bundle\CoreBundle\Entity\Task $task)
    {
        $this->task[] = $task;

        return $this;
    }

    /**
     * Remove task
     *
     * @param \Todo\Bundle\CoreBundle\Entity\Task $task
     */
    public function removeTask(\Todo\Bundle\CoreBundle\Entity\Task $task)
    {
        $this->task->removeElement($task);
    }

    /**
     * Get task
     *
     * @return \Doctrine\Common\Collections\Collection
     */
    public function getTask()
    {
        return $this->task;
    }
}
```

Comment.php

```
<?php

namespace Todo\Bundle\CoreBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Comment
 *
 * @ORM\Table(name="comment")
 * @ORM\Entity
 */
class Comment
{
    /**
```

```
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="comment", type="text", nullable=true)
     */
    private $comment;

    /**
     * @var string
     *
     * @ORM\Column(name="changed", type="text", nullable=true)
     */
    private $changed;

    /**
     * @var \Task
     *
     * @ORM\ManyToOne(targetEntity="Task")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="task_id", referencedColumnName="id")
     * })
     */
    private $task;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="created_at", type="datetime", nullable=false)
     */
    private $createdAt;


    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set comment
     *
     * @param string $comment
     * @return Comment
     */
    public function setComment($comment)
    {
        $this->comment = $comment;

        return $this;
    }

    /**
```

```
     * Get comment
     *
     * @return string
     */
    public function getComment()
    {
        return $this->comment;
    }

    /**
     * Set changed
     *
     * @param string $changed
     * @return Comment
     */
    public function setChanged($changed)
    {
        $this->changed = $changed;

        return $this;
    }

    /**
     * Get changed
     *
     * @return string
     */
    public function getChanged()
    {
        return $this->changed;
    }

    /**
     * Set task
     *
     * @param \Todo\Bundle\CoreBundle\Entity\Task $task
     * @return Comment
     */
    public function setTask(\Todo\Bundle\CoreBundle\Entity\Task $task =
null)
    {
        $this->task = $task;

        return $this;
    }

    /**
     * Get task
     *
     * @return \Todo\Bundle\CoreBundle\Entity\Task
     */
    public function getTask()
    {
        return $this->task;
    }

    /**
     * Set createdAt
     *
     * @param \DateTime $createdAt
     * @return Comment
     */
    public function setCreatedAt($createdAt)
    {
```

```
        $this->createdAt = $createdAt;

        return $this;
    }

    /**
     * Get createdAt
     *
     * @return \DateTime
     */
    public function getCreatedAt()
    {
        return $this->createdAt;
    }
}
```

CustomField.php

```
<?php

namespace Todo\Bundle\CoreBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * CustomField
 *
 * @ORM\Table(name="custom_field")
 * @ORM\Entity
 */
class CustomField
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=45, nullable=false)
     */
    private $key;

    /**
     * @var string
     *
     * @ORM\Column(name="value", type="text", nullable=true)
     */
    private $value;

    /**
     * @ORM\ManyToOne(targetEntity="Task", inversedBy="customFields")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="task_id", referencedColumnName="id")
     * })
     */
    private $task;
```

```
    /**
     * @var \FieldType
     *
     * @ORM\ManyToOne(targetEntity="FieldType")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="field_type_id", referencedColumnName="id")
     * })
     */
    private $fieldType;



    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set key
     *
     * @param string $key
     * @return CustomField
     */
    public function setKey($key)
    {
        $this->key = $key;

        return $this;
    }

    /**
     * Get key
     *
     * @return string
     */
    public function getKey()
    {
        return $this->key;
    }

    /**
     * Set value
     *
     * @param string $value
     * @return CustomField
     */
    public function setValue($value)
    {
        $this->value = $value;

        return $this;
    }

    /**
     * Get value
     *
     * @return string
     */
    public function getValue()
```

```
    {
        return $this->value;
    }

    /**
     * Set task
     *
     * @param \Todo\Bundle\CoreBundle\Entity\Task $task
     * @return CustomField
     */
    public function setTask(\Todo\Bundle\CoreBundle\Entity\Task $task =
null)
    {
        $this->task = $task;

        return $this;
    }

    /**
     * Get task
     *
     * @return \Todo\Bundle\CoreBundle\Entity\Task
     */
    public function getTask()
    {
        return $this->task;
    }

    /**
     * Set fieldType
     *
     * @param \Todo\Bundle\CoreBundle\Entity\FieldType $fieldType
     * @return CustomField
     */
    public function setFieldType(\Todo\Bundle\CoreBundle\Entity\FieldType
$fieldType = null)
    {
        $this->fieldType = $fieldType;

        return $this;
    }

    /**
     * Get fieldType
     *
     * @return \Todo\Bundle\CoreBundle\Entity\FieldType
     */
    public function getFieldType()
    {
        return $this->fieldType;
    }
}
```

FieldType.php

```
<?php

namespace Todo\Bundle\CoreBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * FieldType
```

```
 *
 * @ORM\Table(name="field_type")
 * @ORM\Entity
 */
class FieldType
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="data_type", type="string", length=45,
nullable=false)
     */
    private $dataType;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=45, nullable=false)
     */
    private $name;

    public function __toString() {
        return $this->name;
    }

    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set dataType
     *
     * @param string $dataType
     * @return FieldType
     */
    public function setDataType($dataType)
    {
        $this->dataType = $dataType;

        return $this;
    }

    /**
     * Get dataType
     *
     * @return string
     */
    public function getDataType()
    {
```

```
        return $this->dataType;
    }

    /**
     * Set name
     *
     * @param string $name
     * @return FieldType
     */
    public function setName($name)
    {
        $this->name = $name;

        return $this;
    }

    /**
     * Get name
     *
     * @return string
     */
    public function getName()
    {
        return $this->name;
    }
}
```

Status.php

```
<?php

namespace Todo\Bundle\CoreBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Status
 *
 * @ORM\Table(name="status")
 * @ORM\Entity
 */
class Status
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=45, nullable=false)
     */
    private $name;

    /**
     * @var boolean
     *
     * @ORM\Column(name="active", type="boolean", nullable=false)
```

```
     */
    private $active;

    public function __toString() {
        return $this->name;
    }

    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set name
     *
     * @param string $name
     * @return Status
     */
    public function setName($name)
    {
        $this->name = $name;

        return $this;
    }

    /**
     * Get name
     *
     * @return string
     */
    public function getName()
    {
        return $this->name;
    }

    /**
     * Set active
     *
     * @param boolean $active
     * @return Status
     */
    public function setActive($active)
    {
        $this->active = $active;

        return $this;
    }

    /**
     * Get active
     *
     * @return boolean
     */
    public function getActive()
    {
        return $this->active;
    }
}
```

Task.php

```
<?php

namespace Todo\Bundle\CoreBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use Doctrine\Common\Collections\ArrayCollection;

/**
 * Task
 *
 * @ORM\Table(name="task")
 *
@ORM\Entity(repositoryClass="Todo\Bundle\CoreBundle\Repository\TaskReposito
ry")
 */
class Task
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="description", type="text", nullable=true)
     */
    private $description;

    /**
     * @var string
     *
     * @ORM\Column(name="title", type="string", length=45, nullable=false)
     */
    private $title;

    /**
     * @var integer
     *
     * @ORM\Column(name="progress", type="smallint", nullable=false)
     */
    private $progress;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="event_time", type="datetime", nullable=true)
     */
    private $eventTime;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="creation time", type="datetime", nullable=false)
     */
    private $creationTime;
```

```
    /**
     * @var \User
     *
     * @ORM\ManyToOne(targetEntity="User")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="user_id", referencedColumnName="id")
     * })
     */
    private $user;

    /**
     * @ORM\OneToMany(mappedBy="task", targetEntity="CustomField",
cascade={"persist", "remove"})
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(referencedColumnName="id")
     * })
     */
    private $customField;

    /**
     * @var \Category
     *
     * @ORM\ManyToOne(targetEntity="Category")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="category_id", referencedColumnName="id")
     * })
     */
    private $category;

    /**
     * @var \Status
     *
     * @ORM\ManyToOne(targetEntity="Status")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="status_id", referencedColumnName="id")
     * })
     */
    private $status;

    /**
     * @ORM\OneToMany(mappedBy="task", targetEntity="Comment",
cascade={"persist", "remove"})
     */
    private $comments;

    public function __construct() {
        $this->progress = 0;
        $this->creationTime = new \DateTime();
        $this->customField = new ArrayCollection();
    }

    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set description
     *
```

```
     * @param string $description
     * @return Task
     */
    public function setDescription($description)
    {
        $this->description = $description;

        return $this;
    }

    /**
     * Get description
     *
     * @return string
     */
    public function getDescription()
    {
        return $this->description;
    }

    /**
     * Set title
     *
     * @param string $title
     * @return Task
     */
    public function setTitle($title)
    {
        $this->title = $title;

        return $this;
    }

    /**
     * Get title
     *
     * @return string
     */
    public function getTitle()
    {
        return $this->title;
    }

    /**
     * Set progress
     *
     * @param integer $progress
     * @return Task
     */
    public function setProgress($progress)
    {
        $this->progress = $progress;

        return $this;
    }

    /**
     * Get progress
     *
     * @return integer
     */
    public function getProgress()
    {
        return $this->progress;
```

```
    }

    /**
     * Set eventTime
     *
     * @param \DateTime $eventTime
     * @return Task
     */
    public function setEventTime($eventTime)
    {
        $this->eventTime = $eventTime;

        return $this;
    }

    /**
     * Get eventTime
     *
     * @return \DateTime
     */
    public function getEventTime()
    {
        return $this->eventTime;
    }

    /**
     * Set creationTime
     *
     * @param \DateTime $creationTime
     * @return Task
     */
    public function setCreationTime($creationTime)
    {
        $this->creationTime = $creationTime;

        return $this;
    }

    /**
     * Get creationTime
     *
     * @return \DateTime
     */
    public function getCreationTime()
    {
        return $this->creationTime;
    }

    /**
     * Set category
     *
     * @param \Todo\Bundle\CoreBundle\Entity\Category $category
     * @return Task
     */
    public function setCategory(\Todo\Bundle\CoreBundle\Entity\Category
$category = null)
    {
        $this->category = $category;

        return $this;
    }

    /**
     * Get category
```

```
     *
     * @return \Todo\Bundle\CoreBundle\Entity\Category
     */
    public function getCategory()
    {
        return $this->category;
    }

    /**
     * Set status
     *
     * @param \Todo\Bundle\CoreBundle\Entity\Status $status
     * @return Task
     */
    public function setStatus(\Todo\Bundle\CoreBundle\Entity\Status $status
= null)
    {
        $this->status = $status;

        return $this;
    }

    /**
     * Get status
     *
     * @return \Todo\Bundle\CoreBundle\Entity\Status
     */
    public function getStatus()
    {
        return $this->status;
    }

    /**
     * Set user
     *
     * @param \Todo\Bundle\CoreBundle\Entity\User $user
     * @return Task
     */
    public function setUser(\Todo\Bundle\CoreBundle\Entity\User $user =
null)
    {
        $this->user = $user;

        return $this;
    }

    /**
     * Get user
     *
     * @return \Todo\Bundle\CoreBundle\Entity\User
     */
    public function getUser()
    {
        return $this->user;
    }

    /**
     * Add customField
     *
     * @param \Todo\Bundle\CoreBundle\Entity\CustomField $customField
     * @return Task
     */
    public function
addCustomField(\Todo\Bundle\CoreBundle\Entity\CustomField $customField)
```

```
    {
        $this->customField->add($customField);

        return $this;
    }

    /**
     * Add comments
     *
     * @param \Todo\Bundle\CoreBundle\Entity\Comment $comments
     * @return Task
     */
    public function addComment(\Todo\Bundle\CoreBundle\Entity\Comment
$comments)
    {
        $this->comments[] = $comments;

        return $this;
    }

    /**
     * Remove comments
     *
     * @param \Todo\Bundle\CoreBundle\Entity\Comment $comments
     */
    public function removeComment(\Todo\Bundle\CoreBundle\Entity\Comment
$comments)
    {
        $this->comments->removeElement($comments);
    }

    /**
     * Get comments
     *
     * @return \Doctrine\Common\Collections\Collection
     */
    public function getComments()
    {
        return $this->comments;
    }

    /**
     * Remove customField
     *
     * @param \Todo\Bundle\CoreBundle\Entity\CustomField $customField
     */
    public function
removeCustomField(\Todo\Bundle\CoreBundle\Entity\CustomField $customField)
    {
        $this->customField->removeElement($customField);
    }

    /**
     * Get customField
     *
     * @return \Doctrine\Common\Collections\Collection
     */
    public function getCustomField()
    {
        return $this->customField;
    }
}
```

User.php

```
<?php

namespace Todo\Bundle\CoreBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Validator\Constraints as Assert;
use Symfony\Component\Security\Core\User\UserInterface;
use Symfony\Component\Security\Core\User\EquatableInterface;

/**
 * User
 *
 * @ORM\Table(name="user")
 *
@ORM\Entity(repositoryClass="Todo\Bundle\CoreBundle\Repository\UserReposito
ry")
 */
class User implements UserInterface, \Serializable
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="username", type="string", length=25,
nullable=false)
     * @Assert\NotBlank()
     * @Assert\MinLength(
     *     limit=4,
     *     message="Username must have at least {{ limit }} characters."
     * )
     */
    private $username;

    /**
     * @var string
     *
     * @ORM\Column(name="password", type="string", length=45,
nullable=false)
     * @Assert\MinLength(
     *     limit=6,
     *     message="Password must have at least {{ limit }} characters."
     * )
     */
    private $password;

    /**
     * @var string
     *
     * @ORM\Column(name="salt", type="string", length=45, nullable=false)
     */
    private $salt;

    /**
```

```
     * @var string
     *
     * @ORM\Column(name="firstname", type="string", length=45,
nullable=true)
     */
    private $firstname;

    /**
     * @var string
     *
     * @ORM\Column(name="lastname", type="string", length=45,
nullable=true)
     */
    private $lastname;



    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set username
     *
     * @param string $username
     * @return User
     */
    public function setUsername($username)
    {
        $this->username = $username;

        return $this;
    }

    /**
     * Get username
     *
     * @return string
     */
    public function getUsername()
    {
        return $this->username;
    }

    /**
     * Set password
     *
     * @param string $password
     * @return User
     */
    public function setPassword($password)
    {
        $this->password = $password;

        return $this;
    }

    /**
```

```
     * Get password
     *
     * @return string
     */
    public function getPassword()
    {
        return $this->password;
    }

    /**
     * Set salt
     *
     * @param string $salt
     * @return User
     */
    public function setSalt($salt)
    {
        $this->salt = $salt;

        return $this;
    }

    /**
     * Get salt
     *
     * @return string
     */
    public function getSalt()
    {
        return $this->salt;
    }

    /**
     * Set firstname
     *
     * @param string $firstname
     * @return User
     */
    public function setFirstname($firstname)
    {
        $this->firstname = $firstname;

        return $this;
    }

    /**
     * Get firstname
     *
     * @return string
     */
    public function getFirstname()
    {
        return $this->firstname;
    }

    /**
     * Set lastname
     *
     * @param string $lastname
     * @return User
     */
    public function setLastname($lastname)
    {
        $this->lastname = $lastname;
```

```
        return $this;
    }

    /**
     * Get lastname
     *
     * @return string
     */
    public function getLastname()
    {
        return $this->lastname;
    }

    public function getRoles()
    {
        return array(
            'ROLE_USER'
        );
    }

    public function eraseCredentials()
    {
    }

    /**
     * @see \Serializable::serialize()
     */
    public function serialize()
    {
        return serialize(array(
            $this->id,
        ));
    }

    /**
     * @see \Serializable::unserialize()
     */
    public function unserialize($serialized)
    {
        list (
            $this->id,
        ) = unserialize($serialized);
    }

    public function isEnabled()
    {
        return true;
    }

    public function isAccountNonExpired()
    {
        return true;
    }

    public function isAccountNonLocked()
    {
        return true;
    }

    public function isCredentialsNonExpired()
    {
        return true;
    }
```

```
    public function isEqualTo(UserInterface $user)
    {
        return $this->id === $user->getId();
    }
}
```

CategoryType.php

```
<?php

namespace Todo\Bundle\CoreBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolverInterface;

class CategoryType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array
$options)
    {
        $builder
            ->add('name')
        ;
    }

    public function setDefaultOptions(OptionsResolverInterface $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' => 'Todo\Bundle\CoreBundle\Entity\Category'
        ));
    }

    public function getName()
    {
        return 'todo_bundle_corebundle_categorytype';
    }
}
```

CommentType.php

```
<?php

namespace Todo\Bundle\CoreBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolverInterface;

class CommentType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array
$options)
    {
        $builder
            ->add('comment')
        ;
    }

    public function setDefaultOptions(OptionsResolverInterface $resolver)
```

```
    {
        $resolver->setDefaults(array(
            'data_class' => 'Todo\Bundle\CoreBundle\Entity\Comment'
        ));
    }

    public function getName()
    {
        return 'todo_bundle_corebundle_commenttype';
    }
}
```

CustomFieldType.php

```
<?php

namespace Todo\Bundle\CoreBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolverInterface;

class CustomFieldType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array
$options)
    {
        $builder
            ->add('key', null, array(
                'attr'          => array(
                    'class'         => 'key'
                )
            ))
            ->add('value', 'text', array(
                'attr'          => array(
                    'class'         => 'value'
                )
            ))
            ->add('fieldType', 'entity', array(
                'required' => true,
                'class' => 'TodoCoreBundle:FieldType'
            ))
        ;
    }

    public function setDefaultOptions(OptionsResolverInterface $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' => 'Todo\Bundle\CoreBundle\Entity\CustomField'
        ));
    }

    public function getName()
    {
        return 'todo_bundle_corebundle_customfieldtype';
    }
}
```

TaskType.php

```
<?php
```

```php
namespace Todo\Bundle\CoreBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolverInterface;
use Doctrine\ORM\EntityRepository;
use Todo\Bundle\CoreBundle\Entity\User;
use Todo\Bundle\CoreBundle\Form\CustomFieldType;

class TaskType extends AbstractType
{
    /**
     * @var User $user
     */
    protected $user;

    public function __construct(User $user) {
        $this->user = $user;
    }

    public function buildForm(FormBuilderInterface $builder, array
$options)
    {
        $user = $this->user;

        $builder
            ->add('title')
            ->add('status', 'entity', array(
                'required' => true,
                'empty_value' => false,
                'class' => 'TodoCoreBundle:Status',
            ))
            ->add('description')
            ->add('progress', 'choice', array(
                'choices'  => array(
                    0 => '0%',
                    5 => '5%',
                    10 => '10%',
                    15 => '15%',
                    20 => '20%',
                    25 => '25%',
                    30 => '30%',
                    35 => '35%',
                    40 => '40%',
                    45 => '45%',
                    50 => '50%',
                    55 => '55%',
                    60 => '60%',
                    65 => '65%',
                    70 => '70%',
                    75 => '75%',
                    80 => '80%',
                    85 => '85%',
                    90 => '90%',
                    95 => '95%',
                    1000 => '100%'
                ),
            ))
            ->add('eventTime', null, array(
                'widget'        => 'single_text',
                'attr'          => array(
                    'class'         => 'datepicker'
                ),
```

```
                'format' => 'dd-MM-yyyy h:m'
            ))
            ->add('category', 'entity', array(
                'required' => true,
                'class' => 'TodoCoreBundle:Category',
                    'query_builder' => function(EntityRepository $er) use
($user) {
                        return $er->createQueryBuilder('c')
                            ->where('c.user = :user')
                            ->setParameter('user', $user)
                            ->orderBy('c.name', 'ASC');
            }))
            ->add('customField', 'collection', array(
                'type'          => new CustomFieldType(),
                'allow_add'     => true,
            ))
        ;
    }

    public function setDefaultOptions(OptionsResolverInterface $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' => 'Todo\Bundle\CoreBundle\Entity\Task'
        ));
    }

    public function getName()
    {
        return 'todo_bundle_corebundle_tasktype';
    }
}
```

UserType.php

```
<?php

namespace Todo\Bundle\CoreBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolverInterface;

class UserType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array
$options)
    {
        $builder
            ->add('username')
            ->add('password', 'password')
            ->add('firstname')
            ->add('lastname')
        ;
    }

    public function setDefaultOptions(OptionsResolverInterface $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' => 'Todo\Bundle\CoreBundle\Entity\User'
        ));
    }

    public function getName()
```

```
    {
        return 'todo_bundle_corebundle_usertype';
    }
}
```

UserManager.php

```
<?php

namespace Todo\Bundle\CoreBundle\Manager;

use Todo\Bundle\CoreBundle\Entity\User;
use Symfony\Component\Security\Core\Encoder\EncoderFactory;

class UserManager
{
    /**
     * @param User $user
     * @param string $password
     * @param EncoderFactory $ef
     */
    public function setPassword($user, $password, EncoderFactory $ef)
    {
        $salt = md5(uniqid());
        $user->setSalt($salt);

        $encoder = $ef->getEncoder($user);
        $user->setPassword($encoder->encodePassword($password, $salt));
    }
}
```

TaskRepository.php

```
<?php

namespace Todo\Bundle\CoreBundle\Repository;

use Doctrine\ORM\EntityRepository;
use Todo\Bundle\CoreBundle\Entity\User;
use Todo\Bundle\CoreBundle\Entity\Category;

/**
 * TaskRepository
 *
 * This class was generated by the Doctrine ORM. Add your own custom
 * repository methods below.
 */
class TaskRepository extends EntityRepository
{
    public function getByUserAndCategory(User $user, Category $category)
    {
        return $this->createQueryBuilder('t')
                ->where('t.user = :user')
                ->andWhere('t.category = :category')
                ->orderBy('t.eventTime')
                ->setParameters(array(
                    'category'  => $category,
                    'user'      => $user
                ))
                ->getQuery()
                ->getResult();
```

```
    }
}
```

UserRepository.php

```
<?php

namespace Todo\Bundle\CoreBundle\Repository;

use Doctrine\ORM\EntityRepository;
use Symfony\Component\Security\Core\User\UserProviderInterface;
use Symfony\Component\Security\Core\User\UserInterface;

/**
 * UserRepository
 *
 * This class was generated by the Doctrine ORM. Add your own custom
 * repository methods below.
 */
class UserRepository extends EntityRepository implements
UserProviderInterface
{
    public function loadUserByUsername($username)
    {
        // nolasa no datu bāzes lietotāju ar padoto lietotājvārdu
        $q = $this
            ->createQueryBuilder('u')
            ->where('u.username = :username')
            ->setParameter('username', $username)
            ->getQuery()
        ;

        return $q->getSingleResult();
    }

    public function refreshUser(UserInterface $user)
    {
        $class = get_class($user);
        if (!$this->supportsClass($class)) {
            throw new UnsupportedUserException(
                sprintf(
                    'Instances of "%s" are not supported.',
                    $class
                )
            );
        }

        return $this->find($user->getId());
    }

    public function supportsClass($class)
    {
        return $this->getEntityName() === $class
            || is_subclass_of($class, $this->getEntityName());
    }
}
```

SecurityController.php

```
<?php
```

```php
namespace Todo\SecurityBundle\Controller;

use Todo\Bundle\ClientBundle\Controller\CoreController;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;
use Symfony\Component\HttpFoundation\Response;
use Todo\Bundle\CoreBundle\Form\UserType;
use Symfony\Component\Security\Core\SecurityContext;
use Symfony\Component\Security\Core\Exception\BadCredentialsException;
use Todo\Bundle\CoreBundle\Entity\Category;

class SecurityController extends CoreController
{
    /**
     * @Route("/login-check", name="todo_login_check")
     */
    public function logonCheckAction()
    {

    }

    /**
     * @Route("/login", name="todo_login")
     */
    public function logonAction()
    {
        $request = $this->getRequest();
        $session = $request->getSession();

        // get the login error if there is one
        if ($request->attributes-
>has(SecurityContext::AUTHENTICATION_ERROR)) {
            $error = $request->attributes->get(
                SecurityContext::AUTHENTICATION_ERROR
            );
        } else {
            $error = $session->get(SecurityContext::AUTHENTICATION_ERROR);
            $session->remove(SecurityContext::AUTHENTICATION_ERROR);
        }

        $registrationForm = $this->createForm(new UserType());

        return $this->render(
            'TodoClientBundle:Homepage:index.html.twig',
            array(
                // last username entered by the user
                'last_username' => $session-
>get(SecurityContext::LAST_USERNAME),
                'error'         => $error,
                'registration'  => $registrationForm->createView()
            )
        );
    }

    /**
     * @Route("/signup", name="todo signup")
     * @Method("POST")
     */
    public function signupAction()
    {
        $request = $this->getRequest();
        $view = 'TodoClientBundle:Homepage:index.html.twig';

        $form = $this->createForm(new UserType());
```

```
        $form->bind($request);

        if ($form->isValid()) {

            $user = $form->getData();

            $existingUSer = $this->getUserRepository()-
>findOneByUsername($data->getUsername());
            if (!$existingUSer) {
                $um = $this->getUserManager();

                $um->setPassword($user, $user->getPassword(), $this-
>get('security.encoder_factory'));

                $em = $this->getDoctrine()->getEntityManager();

                $em->persist($user);

                $category = new Category();
                $category->setUser($user);
                $category->setName('ToDo');

                $em->persist($category);

                $em->flush();

                $form = $this->createForm(new UserType());

                return $this->render($view, array(
                    'registration'  => $form->createView(),
                    'success'       => true
                ));
            }

            return $this->render($view, array(
                'registration'  => $form->createView(),
                'error' => 'Username taken',
            ));
        }

        return $this->render($view, array(
            'registration'  => $form->createView(),
            'error' => 'You have form errors',
        ));
    }
}
```

3. PIELIKUMS IZVEIDOTĀS TĪMEKĻA VIETNES KOD ZF2 IETVARĀ

global.php

```
<?php
// Publiskie datu bÄzes uzstÄdÄ«jumi
return array(
    'db' => array(
        'driver'         => 'Pdo',
        'dsn'            => 'mysql:dbname=todo;host=localhost',
        'driver_options' => array(
            PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES \'UTF8\''
        ),
    ),
    'service_manager' => array(
        'factories' => array(
            'Zend\Db\Adapter\Adapter'
                => 'Zend\Db\Adapter\AdapterServiceFactory',
        ),
    ),
)
```

local.php

```
<?php
// Privātie datu bāzes uzstādījumi
return array(
    'db' => array(
        'username' => 'root',
        'password' => 'messager',
    ),
);
```

application.config.php

```
<?php
// ietvara uzstādījumi
return array(
    'modules' => array(
        'Application',
    ),
    'module_listener_options' => array(
        'module_paths' => array(
            './module',
            './vendor',
        ),
        'config_glob_paths' => array(
            'config/autoload/{,*.}{global,local}.php',
        ),
    ),
);
```

module.config.php

```php
<?php
// Moduļa uzstādījumi
return array(
    'router' => array(
        'routes' => array( // maršruti
            'home' => array(
                'type' => 'Zend\Mvc\Router\Http\Literal',
                'options' => array(
                    'route'    => '/',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Index',
                        'action'     => 'index',
                    ),
                ),
            ),
            'tasks' => array(
                'type' => 'Zend\Mvc\Router\Http\Literal',
                'options' => array(
                    'route'    => '/list',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Task',
                        'action'     => 'index',
                    ),
                ),
            ),
            'task_add' => array(
                'type' => 'Zend\Mvc\Router\Http\Literal',
                'options' => array(
                    'route'    => '/task/add',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Task',
                        'action'     => 'add',
                    ),
                ),
            ),
            'task_update' => array(
                'type' => 'segment',
                'options' => array(
                    'route'    => '/task/update[/:id]',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Task',
                        'action'     => 'update',
                    ),
                ),
            ),
            'task_remove' => array(
                'type' => 'segment',
                'options' => array(
                    'route'    => '/task/remove[/:id]',
                    'constraints' => array(
                        'id'     => '[0-9]+',
                    ),
                    'defaults' => array(
                        'controller' => 'Application\Controller\Task',
                        'action'     => 'remove',
                    ),
                ),
            ),
            'categories' => array(
                'type' => 'Zend\Mvc\Router\Http\Literal',
                'options' => array(
                    'route'    => '/categories',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Category',
```

```
                        'action'     => 'list',
                    ),
                ),
            ),
            'category_add' => array(
                'type' => 'Zend\Mvc\Router\Http\Literal',
                'options' => array(
                    'route'    => '/category/add',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Category',
                        'action'     => 'add',
                    ),
                ),
            ),
            'category_remove' => array(
                'type' => 'segment',
                'options' => array(
                    'route'    => '/category/remove[/:id]',
                    'constraints' => array(
                        'id'     => '[0-9]+',
                    ),
                    'defaults' => array(
                        'controller' => 'Application\Controller\Category',
                        'action'     => 'remove',
                    ),
                ),
            ),
            'login' => array(
                'type' => 'Zend\Mvc\Router\Http\Literal',
                'options' => array(
                    'route'    => '/login',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Index',
                        'action'     => 'doLogin',
                    ),
                ),
            ),
            'logout' => array(
                'type' => 'Zend\Mvc\Router\Http\Literal',
                'options' => array(
                    'route'    => '/logout',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Index',
                        'action'     => 'logout',
                    ),
                ),
            ),
        ),
    ),
    'service_manager' => array( //servisi
        'abstract_factories' => array(
            'Zend\Cache\Service\StorageCacheAbstractServiceFactory',
            'Zend\Log\LoggerAbstractServiceFactory',
        ),
        'aliases' => array(
            'translator' => 'MvcTranslator',
        ),
    ),
    'translator' => array(
        'locale' => 'en_US',
        'translation_file_patterns' => array(
            array(
                'type'     => 'gettext',
                'base_dir' => __DIR__ . '/../language',
```

```
                'pattern'  => '%s.mo',
            ),
        ),
    ),
    'controllers' => array( // kontrolieri
        'invokables' => array(
            'Application\Controller\Index'      =>
'Application\Controller\IndexController',
            'Application\Controller\Category'   =>
'Application\Controller\CategoryController',
            'Application\Controller\Task'       =>
'Application\Controller\TaskController'
        ),
    ),
    'view_manager' => array(
        'display_not_found_reason' => true,
        'display_exceptions'       => true,
        'doctype'                  => 'HTML5',
        'not_found_template'       => 'error/404',
        'exception_template'       => 'error/index',
        'template_map' => array(
            'layout/loggedinLayout'   => __DIR__ .
'/../view/layout/loggedinLayout.phtml',
            'layout/layout'           => __DIR__ .
'/../view/layout/layout.phtml',
            'application/index/index' => __DIR__ .
'/../view/application/index/index.phtml',
            'error/404'               => __DIR__ .
'/../view/error/404.phtml',
            'error/index'             => __DIR__ .
'/../view/error/index.phtml',
        ),
        'template_path_stack' => array(
            __DIR__ . '/../view',
        ),
    ),
);
```

CategoryController.php

```
<?php
namespace Application\Controller;

use Zend\Mvc\Controller\AbstractActionController;
use Zend\View\Model\ViewModel;
use Zend\Session\Container;
use Application\Form\CategoryForm;
use Application\Model\UserTable;
use Application\Model\CategoryTable;
use Application\Model\TaskTable;
use Application\Model\Category;

class CategoryController extends AbstractActionController
{
    protected $userTable;
    protected $categoryTable;
    protected $taskTable;

    // tabulu ielādes metodes
    /**
     * @return UserTable
     */
    public function getUserTable()
```

```
    {
        if (!$this->userTable) {
            $sm = $this->getServiceLocator();
            $this->userTable = $sm->get('Application\Model\UserTable');
        }
        return $this->userTable;
    }

    /**
     * @return CategoryTable
     */
    public function getCategoryTable()
    {
        if (!$this->categoryTable) {
            $sm = $this->getServiceLocator();
            $this->categoryTable = $sm-
>get('Application\Model\CategoryTable');
        }
        return $this->categoryTable;
    }

    /**
     * @return TaskTable
     */
    public function getTaskTable()
    {
        if (!$this->taskTable) {
            $sm = $this->getServiceLocator();
            $this->taskTable = $sm->get('Application\Model\TaskTable');
        }
        return $this->taskTable;
    }

    public function listAction()
    {
        $user_session = new Container('todo_zend_auth');
        $un = $user_session->username;
        if (($user = $this->getUserTable()->getByUsername($un)) === false)
{
            return $this->redirect()->toRoute('home');
        }

        $categories = $this->getCategoryTable()->fetchAllForUser($user);

        $layout = $this->layout();
        $layout->setTemplate('layout/loggedinLayout');

        return new ViewModel(array(
            'categories'    => $categories
        ));
    }

    public function addAction()
    {
        $user_session = new Container('todo_zend_auth');
        $un = $user session->username;
        if (($user = $this->getUserTable()->getByUsername($un)) === false)
{
            return $this->redirect()->toRoute('home');
        }
        $form = new CategoryForm();

        $request = $this->getRequest();
```

```
        if ($request->isPost()) {
            $category = new Category();
            $form->setInputFilter($category->getInputFilter());
            $form->setData($request->getPost());

            if ($form->isValid()) {
                $category->exchangeArray($form->getData());
                $category->user_id = $user->id;

                $this->getCategoryTable()->save($category);

                return $this->redirect()->toRoute('categories', array());
            }
        }

        $layout = $this->layout();
        $layout->setTemplate('layout/loggedinLayout');

        return new ViewModel(array(
            'form'      => $form
        ));
    }

    public function removeAction()
    {
        $user_session = new Container('todo_zend_auth');
        $un = $user_session->username;
        if (($user = $this->getUserTable()->getByUsername($un)) === false)
{
             return $this->redirect()->toRoute('home');
        }

        $id = $this->getEvent()->getRouteMatch()->getParam('id');

        $category = $this->getCategoryTable()->get($id);

        if (!$category) {
            return $this->redirect()->toRoute('categories');
        }

        foreach ($this->getTaskTable()->fetchAllForCategory($category) as
$task) {
            $this->getTaskTable()->delete($task->id);
        }

        $this->getCategoryTable()->delete($id);

        return $this->redirect()->toRoute('categories');
    }

}
```

IndexController.php

```
<?php
namespace Application\Controller;

use Zend\Mvc\Controller\AbstractActionController;
use Zend\View\Model\ViewModel;
use Application\Model\UserTable;
use Application\Form\RegistrationForm;
use Application\Form\LoginForm;
use Application\Model\User;
```

```
use Zend\Session\Container;
use Application\Model\Category;
use Application\Model\CategoryTable;

class IndexController extends AbstractActionController
{
    protected $userTable;
    protected $categoryTable;

    /**
     * @return CategoryTable
     */
    public function getCategoryTable()
    {
        if (!$this->categoryTable) {
            $sm = $this->getServiceLocator();
            $this->categoryTable = $sm-
>get('Application\Model\CategoryTable');
        }
        return $this->categoryTable;
    }

    /**
     * @return UserTable
     */
    public function getUserTable()
    {
        if (!$this->userTable) {
            $sm = $this->getServiceLocator();
            $this->userTable = $sm->get('Application\Model\UserTable');
        }
        return $this->userTable;
    }

    public function indexAction()
    {
        /**
         * is user exista and is logged in
         */
        $user_session = new Container('todo_zend_auth');
        $un = $user_session->username;
        if  ($this->getUserTable()->usernameAvailable($un) === false) {
             return $this->redirect()->toRoute('tasks');
        }

        $form = new RegistrationForm();
        $form->get('submit')->setValue('Add');

        $loginForm = new LoginForm();
        $loginForm->get('submit')->setValue('Add');

        $request = $this->getRequest();
        $succes = false;

        if ($request->isPost()) {
            $user = new User();
            $form->setInputFilter($user->getInputFilter());
            $form->setData($request->getPost());

            if ($form->isValid()) {
                $user->exchangeArray($form->getData());

                if($this->getUserTable()->usernameAvailable($user-
>username)) {
```

```
                    /**
                     * Simulating Symfony2 password generation
                     */
                    $salt = substr(sha1(md5(time())), 0, 32);
                    $user->password = hash('sha1', $user-
>password.'{'.$salt.'}');
                    $user->salt = $salt;

                    $this->getUserTable()->saveUser($user);

                    /**
                     * Refresh user to get id
                     */
                    $user = $this->getUserTable()->getByUsername($user-
>username);

                    $category = new Category();
                    $category->user_id = $user->id;
                    $category->name = 'ToDo';

                    $this->getCategoryTable()->save($category);

                    $succes = true;
                    $form = new RegistrationForm();
                    $form->get('submit')->setValue('Add');
                }
            }
        }

        return new ViewModel(array(
            'loginForm'      => $loginForm,
            'form'       => $form,
            'success'    => $succes
        ));
    }

    public function doLoginAction()
    {
        $request = $this->getRequest();
        $form = new LoginForm();
        $form->get('submit')->setValue('Add');

        if ($request->isPost()) {
            $user = new User();
            $form->setInputFilter($user->getInputFilter());
            $form->setData($request->getPost());

            if ($form->isValid()) {
                $user->exchangeArray($form->getData());
                $foundUser = $this->getUserTable()->getByUsername($user-
>username);

                if ($foundUser !== false) {
                    /**
                     * Simulating Symfony2 password encoding
                     */
                    $salt = $foundUser->salt;
                    $generatedPassword = hash('sha1', $user-
>password.'{'.$salt.'}');

                    if ($generatedPassword == $foundUser->password) {
                        $user_session = new Container('todo_zend_auth');
                        $user_session->username = $user->username;
```

```
                            return $this->redirect()->toRoute('tasks', array(
                            ));
                        }
                    }
                }
            }

            return $this->redirect()->toRoute('home');
        }

        public function logoutAction()
        {
            $user_session = new Container('todo_zend_auth');
            $user_session->username = null;

            return $this->redirect()->toRoute('home');
        }
}
```

TaskController.php

```
<?php
namespace Application\Controller;

use Zend\Mvc\Controller\AbstractActionController;
use Zend\View\Model\ViewModel;
use Application\Model\UserTable;
use Zend\Session\Container;
use Application\Model\CategoryTable;
use Application\Model\TaskTable;
use Application\Model\StatusTable;
use Application\Form\TaskForm;
use Application\Model\Task;

class TaskController extends AbstractActionController
{
    protected $userTable;
    protected $categoryTable;
    protected $taskTable;
    protected $statusTable;

    /**
     * @return UserTable
     */
    public function getUserTable()
    {
        if (!$this->userTable) {
            $sm = $this->getServiceLocator();
            $this->userTable = $sm->get('Application\Model\UserTable');
        }
        return $this->userTable;
    }

    /**
     * @return StatusTable
     */
    public function getStatusTable()
    {
        if (!$this->statusTable) {
            $sm = $this->getServiceLocator();
            $this->statusTable = $sm->get('Application\Model\StatusTable');
        }
```

```
        return $this->statusTable;
    }

    /**
     * @return CategoryTable
     */
    public function getCategoryTable()
    {
        if (!$this->categoryTable) {
            $sm = $this->getServiceLocator();
            $this->categoryTable = $sm-
>get('Application\Model\CategoryTable');
        }
        return $this->categoryTable;
    }

    /**
     * @return TaskTable
     */
    public function getTaskTable()
    {
        if (!$this->taskTable) {
            $sm = $this->getServiceLocator();
            $this->taskTable = $sm->get('Application\Model\TaskTable');
        }
        return $this->taskTable;
    }

    public function indexAction()
    {
        $user_session = new Container('todo_zend_auth');
        $un = $user_session->username;
        if (($user = $this->getUserTable()->getByUsername($un)) === false)
{
             return $this->redirect()->toRoute('home');
        }

        $categories = $this->getCategoryTable()->fetchAllForUser($user);

        $categoryArray = array();

        foreach ($categories as $category) {
            $tasks = array();
            foreach ($this->getTaskTable()->fetchAllForCategory($category)
as $task) {
                $task->status_id = $this->getStatusTable()->get($task-
>status_id);
                $tasks[] = $task;
            }

            $category->tasks = $tasks;
            $categoryArray[] = $category;
        }

        $layout = $this->layout();
        $layout->setTemplate('layout/loggedinLayout');

        return new ViewModel(array(
            'categories'    => $categoryArray
        ));
    }

    public function addAction() {
        $user_session = new Container('todo_zend_auth');
```

```
        $un = $user_session->username;
        if (($user = $this->getUserTable()->getByUsername($un)) === false)
{
             return $this->redirect()->toRoute('home');
        }

        $statuses = $this->getStatusTable()->fetchAll();
        $categories = $this->getCategoryTable()->fetchAllForUser($user);

        $form = new TaskForm($categories, $statuses);

        $request = $this->getRequest();
        if ($request->isPost()) {
            $task = new Task();
            $form->setInputFilter($task->getInputFilter());
            $form->setData($request->getPost());

            if ($form->isValid()) {
                $task->exchangeArray($form->getData());
                $task->user_id = $user->id;
                $task->creation_time = new \DateTime();
                if (strlen($task->event_time) > 10) {
                    $task->event_time = new \DateTime($task->event_time);
                } else {
                    $task->event_time = null;
                }

                $this->getTaskTable()->save($task);

                return $this->redirect()->toRoute('tasks', array());
            }
        }

        $layout = $this->layout();
        $layout->setTemplate('layout/loggedinLayout');

        return new ViewModel(array(
            'form'      => $form
        ));
    }

    public function updateAction() {
        $user_session = new Container('todo_zend_auth');
        $un = $user_session->username;
        if (($user = $this->getUserTable()->getByUsername($un)) === false)
{
             return $this->redirect()->toRoute('home');
        }

        $id = (int) $this->params()->fromRoute('id', 0);
        if (!$id) {
            return $this->redirect()->toRoute('task_add');
        }
        $task = $this->getTaskTable()->get($id);

        $statuses = $this->getStatusTable()->fetchAll();
        $categories = $this->getCategoryTable()->fetchAllForUser($user);

        $form = new TaskForm($categories, $statuses);
        $form->bind($task);

        $request = $this->getRequest();
        if ($request->isPost()) {
            $form->setInputFilter($task->getInputFilter());
```

```
            $form->setData($request->getPost());

            if ($form->isValid()) {
                $task = $form->getData();
                if (strlen($task->event_time) > 10) {
                    $task->event_time = new \DateTime($task->event_time);
                } else {
                    $task->event_time = null;
                }

                $task->user_id = $user->id;

                $this->getTaskTable()->save($task);

                return $this->redirect()->toRoute('tasks', array());
            }
        }

        $layout = $this->layout();
        $layout->setTemplate('layout/loggedinLayout');

        return new ViewModel(array(
            'form'      => $form,
            'id'        => $id
        ));
    }

    public function removeAction() {
        $user_session = new Container('todo_zend_auth');
        $un = $user_session->username;
        if (($user = $this->getUserTable()->getByUsername($un)) === false)
{
             return $this->redirect()->toRoute('home');
        }

        $id = (int) $this->params()->fromRoute('id', 0);
        if ($id) {
            $this->getTaskTable()->delete($id);
        }

        return $this->redirect()->toRoute('tasks');
    }
}
```

CategoryForm.php

```
<?php
namespace Application\Form;

use Zend\Form\Form;

class CategoryForm extends Form
{
    public function __construct($name = null)
    {
        // padotais parametrs tiie ignorēts jo šī klase ir domātatieši
priekš kategorijām
        parent::__construct('category');
        $this->setAttribute('method', 'post');
        $this->add(array(
            'name' => 'name',
            'attributes' => array(
                'type'  => 'text',
```

```
            ),
            'options' => array(
                'label' => 'Username',
            ),
        ));
        $this->add(array(
            'name' => 'submit',
            'attributes' => array(
                'type'  => 'submit',
                'value' => 'Submit',
                'class' => 'btn btn-success',
                'id' => 'submitbutton',
            ),
        ));
    }
}
```

LoginForm.php

```
<?php
namespace Application\Form;

use Zend\Form\Form;

class LoginForm extends Form
{
    public function __construct($name = null)
    {
        // we want to ignore the name passed
        parent::__construct('user');
        $this->setAttribute('method', 'post');
        $this->add(array(
            'name' => 'username',
            'attributes' => array(
                'type'  => 'text',
            ),
            'options' => array(
                'label' => 'Username',
            ),
        ));
        $this->add(array(
            'name' => 'password',
            'attributes' => array(
                'type'  => 'password',
            ),
            'options' => array(
                'label' => 'Password',
            ),
        ));
        $this->add(array(
            'name' => 'submit',
            'attributes' => array(
                'type'  => 'submit',
                'value' => 'Register',
                'class' => 'btn btn-success',
                'id' => 'submitbutton',
            ),
        ));
    }
}
```

RegistrationForm.php

```php
<?php
namespace Application\Form;

use Zend\Form\Form;

class RegistrationForm extends Form
{
    public function __construct($name = null)
    {
        // we want to ignore the name passed
        parent::__construct('user');
        $this->setAttribute('method', 'post');
        $this->add(array(
            'name' => 'id',
            'attributes' => array(
                'type'  => 'hidden',
            ),
        ));
        $this->add(array(
            'name' => 'username',
            'attributes' => array(
                'type'  => 'text',
            ),
            'options' => array(
                'label' => 'Username',
            ),
        ));
        $this->add(array(
            'name' => 'password',
            'attributes' => array(
                'type'  => 'password',
            ),
            'options' => array(
                'label' => 'Password',
            ),
        ));
        $this->add(array(
            'name' => 'firstname',
            'attributes' => array(
                'type'  => 'text',
            ),
            'options' => array(
                'label' => 'First name',
            ),
        ));
        $this->add(array(
            'name' => 'lastname',
            'attributes' => array(
                'type'  => 'text',
            ),
            'options' => array(
                'label' => 'Last name',
            ),
        ));
        $this->add(array(
            'name' => 'submit',
            'attributes' => array(
                'type'  => 'submit',
                'value' => 'Register',
                'class' => 'btn btn-success',
                'id' => 'submitbutton',
            ),
        ));
    }
```

```
}
```

TaskForm.php

```
<?php
namespace Application\Form;

use Zend\Form\Form;

class TaskForm extends Form
{
    public function __construct($categories, $statuses, $name = null)
    {
        // we want to ignore the name passed
        parent::__construct('task');
        $this->setAttribute('method', 'post');
        $this->add(array(
            'name' => 'id',
            'attributes' => array(
                'type'  => 'hidden',
            ),
        ));
        $this->add(array(
            'name' => 'title',
            'attributes' => array(
                'type'  => 'text',
            ),
            'options' => array(
                'label' => 'Title',
            ),
        ));
        $this->add(array(
            'name' => 'description',
            'attributes' => array(
                'type'  => 'text',
            ),
            'options' => array(
                'label' => 'Description',
            ),
        ));
        $this->add(array(
            'name' => 'progress',
            'type'  => 'Zend\Form\Element\Select',
            'attributes' =>  array(
                'options' => array(
                    0   => '0%',
                    5   => '5%',
                    10   => '10%',
                    15   => '15%',
                    20   => '20%',
                    25   => '25%',
                    30   => '30%',
                    35   => '35%',
                    40   => '40%',
                    45   => '45%',
                    50   => '50%',
                    55   => '55%',
                    60   => '60%',
                    65   => '65%',
                    70   => '70%',
                    75   => '75%',
                    80   => '80%',
                    85   => '85%',
```

```
                    90   => '90%',
                    95   => '95%',
                    100   => '100%',
                )
            ),
            'options' => array(
                'label' => 'Progress',
            ),
        ));
        $this->add(array(
            'name' => 'event_time',
            'attributes' => array(
                'type'  => 'text',
                'class'  => 'datepicker',
            ),
            'options' => array(
                'label' => 'Event Time',
            ),
        ));

        $catData = array();
        foreach ($categories as $category) {
            $catData[$category->id] = $category->name;
        }

        $this->add(array(
            'name' => 'category_id',
            'type'  => 'Zend\Form\Element\Select',
            'attributes' =>  array(
                'options' => $catData
            ),
            'options' => array(
                'label' => 'Category',
            ),
        ));

        $staData = array();
        foreach ($statuses as $status) {
            $staData[$status->id] = $status->name;
        }
        $this->add(array(
            'name' =>   'status_id',
            'type'  => 'Zend\Form\Element\Select',
            'attributes' =>  array(
                'options' => $staData
            ),
            'options' => array(
                'label' => 'Status',
            ),
        ));
        $this->add(array(
            'name' => 'submit',
            'attributes' => array(
                'type'  => 'submit',
                'value' => 'Save',
                'class' => 'btn btn-success',
                'id' => 'submitbutton',
            ),
        ));
    }
}
```

Category.php

```php
<?php
namespace Application\Model;

use Zend\InputFilter\Factory as InputFactory;
use Zend\InputFilter\InputFilter;
use Zend\InputFilter\InputFilterAwareInterface;
use Zend\InputFilter\InputFilterInterface;

class Category implements InputFilterAwareInterface
{
    public $id;
    public $name;
    public $user_id;
    public $tasks;
    protected $inputFilter;

    public function exchangeArray($data)
    {
        $this->id       = (isset($data['id']))       ? $data['id']       :
null;
        $this->name     = (isset($data['name']))     ? $data['name']     :
null;
        $this->user_id  = (isset($data['user_id'])) ? $data['user_id']  :
null;
    }

    public function setInputFilter(InputFilterInterface $inputFilter)
    {
        throw new \Exception("Not used");
    }

    public function getInputFilter()
    {
        if (!$this->inputFilter) {
            $inputFilter = new InputFilter();
            $factory     = new InputFactory();

            $inputFilter->add($factory->createInput(array(
                'name'     => 'id',
                'required' => true,
                'filters'  => array(
                    array('name' => 'Int'),
                ),
            )));

            $inputFilter->add($factory->createInput(array(
                'name'     => 'name',
                'required' => true,
                'filters'  => array(
                    array('name' => 'StripTags'),
                    array('name' => 'StringTrim'),
                ),
                'validators' => array(
                    array(
                        'name'    => 'StringLength',
                        'options' => array(
                            'encoding' => 'UTF-8',
                            'min'      => 4,
                            'max'      => 100,
                        ),
                    ),
                ),
            )));
```

```
            $this->inputFilter = $inputFilter;
        }

        return $this->inputFilter;
    }
}
```

CategoryTable.php

```
<?php
namespace Application\Model;

use Zend\Db\TableGateway\TableGateway;
use Zend\Db\ResultSet\ResultSet;

class CategoryTable
{
    protected $gate;

    public function __construct(TableGateway $gate)
    {
        $this->gate = $gate;
    }

    public function fetchAll()
    {
        $resultSet = $this->gate->select();
        return $resultSet;
    }

    /**
     * @param \Application\Model\User $user
     * @return ResultSet
     */
    public function fetchAllForUser(User $user)
    {
        $resultSet = $this->gate->select(array(
            'user_id'    => (int)$user->id
        ));

        return $resultSet;
    }

    public function get($id)
    {
        $id  = (int) $id;
        $rowset = $this->gate->select(array('id' => $id));
        $row = $rowset->current();
        if (!$row) {
            throw new \Exception("Could not find row $id");
        }
        return $row;
    }

    public function save(Category $model)
    {
        $data = array(
            'name'       => $model->name,
            'user_id'    => $model->user_id,
        );

        $id = (int)$model->id;
        if ($id == 0) {
```

```
            $this->gate->insert($data);
        } else {
            if ($this->get($id)) {
                $this->gate->update($data, array('id' => $id));
            } else {
                throw new \Exception('Form id does not exist');
            }
        }
    }

    public function delete($id)
    {
        $this->gate->delete(array('id' => $id));
    }
}
```

Status.php

```
<?php
namespace Application\Model;

class Status
{
    public $id;
    public $name;
    public $active;

    public function exchangeArray($data)
    {
        $this->id       = (isset($data['id']))       ? $data['id']       :
null;
        $this->name     = (isset($data['name']))     ? $data['name']     :
null;
        $this->active   = (isset($data['active']))   ? $data['active']   :
null;
    }
}
```

StatusTable.php

```
<?php
namespace Application\Model;

use Zend\Db\TableGateway\TableGateway;
use Zend\Db\ResultSet\ResultSet;

class StatusTable
{
    protected $gate;

    public function __construct(TableGateway $gate)
    {
        $this->gate = $gate;
    }

    public function fetchAll()
    {
        $resultSet = $this->gate->select();
        return $resultSet;
    }
```

```
    /**
     * @param \Application\Model\User $user
     * @return ResultSet
     */
    public function fetchAllForUser(User $user)
    {
        $resultSet = $this->gate->select(array(
            'user_id'   => (int)$user->id
        ));

        return $resultSet;
    }

    public function get($id)
    {
        $id  = (int) $id;
        $rowset = $this->gate->select(array('id' => $id));
        $row = $rowset->current();
        if (!$row) {
            throw new \Exception("Could not find row $id");
        }
        return $row;
    }

    public function save(Status $model)
    {
        $data = array(
            'name'      => $model->name,
            'active'    => $model->active,
        );

        $id = (int)$model->id;
        if ($id == 0) {
            $this->gate->insert($data);
        } else {
            if ($this->get($id)) {
                $this->gate->update($data, array('id' => $id));
            } else {
                throw new \Exception('Form id does not exist');
            }
        }
    }

    public function delete($id)
    {
        $this->gate->delete(array('id' => $id));
    }
}
```

Task.php

```
<?php
namespace Application\Model;

use Zend\InputFilter\Factory as InputFactory;
use Zend\InputFilter\InputFilter;
use Zend\InputFilter\InputFilterAwareInterface;
use Zend\InputFilter\InputFilterInterface;

class Task implements InputFilterAwareInterface
{
    public $id;
    public $title;
```

```
    public $description;
    public $progress;
    public $event_time;
    public $creation_time; /* Before save!! */
    public $user_id;
    public $category_id;
    public $status_id;
    protected $inputFilter;

    public function exchangeArray($data)
    {
        $this->id               = (isset($data['id']))               ?
$data['id']               : null;
        $this->title            = (isset($data['title']))            ?
$data['title']            : null;
        $this->description      = (isset($data['description']))      ?
$data['description']      : null;
        $this->progress         = (isset($data['progress']))         ?
$data['progress']         : null;
        $this->event_time       = (isset($data['event_time']))       ?
$data['event_time']       : null;
        $this->creation_time    = (isset($data['creation_time']))    ?
$data['creation_time']    : null;
        $this->user_id          = (isset($data['user_id']))          ?
$data['user_id']          : null;
        $this->category_id      = (isset($data['category_id']))      ?
$data['category_id']      : null;
        $this->status_id        = (isset($data['status_id']))        ?
$data['status_id']        : null;
    }

    public function setInputFilter(InputFilterInterface $inputFilter)
    {
        throw new \Exception("Not used");
    }

    public function getArrayCopy()
    {
        return get_object_vars($this);
    }

    public function getInputFilter()
    {
        if (!$this->inputFilter) {
            $inputFilter = new InputFilter();
            $factory     = new InputFactory();

            $inputFilter->add($factory->createInput(array(
                'name'     => 'id',
                'required' => true,
                'filters'  => array(
                    array('name' => 'Int'),
                ),
            )));

            $inputFilter->add($factory->createInput(array(
                'name'     => 'title',
                'required' => true,
                'filters'  => array(
                    array('name' => 'StripTags'),
                    array('name' => 'StringTrim'),
                ),
                'validators' => array(
                    array(
```

```
                        'name'    => 'StringLength',
                        'options' => array(
                            'encoding' => 'UTF-8',
                            'min'      => 4,
                            'max'      => 100,
                        ),
                    ),
                ),
            )));
            $inputFilter->add($factory->createInput(array(
                'name'     => 'progress',
                'required' => true,
                'filters'  => array(
                    array('name' => 'Int'),
                ),
                'validators' => array(
                  array(
                      'name' => 'Between',
                      'options' => array(
                          'min' => 0,
                          'max' => 100,
                      ),
                  ),
                )
            )));
            $inputFilter->add($factory->createInput(array(
                'name'     => 'category_id',
                'required' => true,
                'filters'  => array(
                    array('name' => 'Int'),
                ),
                'validators' => array(
                  array(
                      'name' => 'Between',
                      'options' => array(
                          'min' => 1,
                          'max' => 100000000,
                      ),
                  ),
                )
            )));
            $inputFilter->add($factory->createInput(array(
                'name'     => 'status_id',
                'required' => true,
                'filters'  => array(
                    array('name' => 'Int'),
                ),
                'validators' => array(
                  array(
                      'name' => 'Between',
                      'options' => array(
                          'min' => 1,
                          'max' => 100000000,
                      ),
                  ),
                )
            )));
            $this->inputFilter = $inputFilter;
        }

        return $this->inputFilter;
    }
}
```

TaskTable.php

```
<?php
namespace Application\Model;

use Zend\Db\TableGateway\TableGateway;
use Zend\Db\ResultSet\ResultSet;

class TaskTable
{
    protected $gate;

    public function __construct(TableGateway $gate)
    {
        $this->gate = $gate;
    }

    public function fetchAll()
    {
        $resultSet = $this->gate->select();
        return $resultSet;
    }

    /**
     * @param \Application\Model\User $user
     * @return ResultSet
     */
    public function fetchAllForUser(User $user)
    {
        $resultSet = $this->gate->select(array(
            'user_id'    => (int)$user->id
        ));

        return $resultSet;
    }

    /**
     * @param \Application\Model\User $category
     * @return ResultSet
     */
    public function fetchAllForCategory(Category $category)
    {
        $resultSet = $this->gate->select(array(
            'category_id'    => (int)$category->id
        ));

        return $resultSet;
    }

    public function get($id)
    {
        $id  = (int) $id;
        $rowset = $this->gate->select(array('id' => $id));
        $row = $rowset->current();
        if (!$row) {
            throw new \Exception("Could not find row $id");
        }
        return $row;
    }

    public function save(Task $model)
    {
        $cls = new \ReflectionClass("DateTime");
```

```
        if ($cls->isInstance( (object) $model->event_time )) {
            $model->event_time = $model->event_time->format('Y-m-d H:i:s');
        }
        if ($cls->isInstance( (object) $model->creation_time )) {
            $model->creation_time = $model->creation_time->format('Y-m-d
H:i:s');
        }

        $data = array(
            'title'         => $model->title,
            'description'   => $model->description,
            'progress'      => $model->progress,
            'event_time'    => $model->event_time,
            'creation_time' => $model->creation_time,
            'user_id'       => $model->user_id,
            'category_id'   => $model->category_id,
            'status_id'     => $model->status_id,
        );

        $id = (int)$model->id;
        if ($id == 0) {
            $this->gate->insert($data);
        } else {
            if ($this->get($id)) {
                $this->gate->update($data, array('id' => $id));
            } else {
                throw new \Exception('Form id does not exist');
            }
        }
    }

    public function delete($id)
    {
        $this->gate->delete(array('id' => $id));
    }
}
```

User.php

```
<?php
namespace Application\Model;

use Zend\InputFilter\Factory as InputFactory;
use Zend\InputFilter\InputFilter;
use Zend\InputFilter\InputFilterAwareInterface;
use Zend\InputFilter\InputFilterInterface;

class User implements InputFilterAwareInterface
{
    public $id;
    public $username;
    public $password;
    public $salt;
    public $firstname;
    public $lastname;
    protected $inputFilter;

    public function exchangeArray($data)
    {
        $this->id           = (isset($data['id']))        ? $data['id']
: null;
        $this->username     = (isset($data['username']))  ?
$data['username']     : null;
```

```
        $this->password     = (isset($data['password']))  ?
$data['password']     : null;
        $this->salt         = (isset($data['salt']))      ? $data['salt']
: null;
        $this->firstname    = (isset($data['firstname'])) ?
$data['firstname']    : null;
        $this->lastname     = (isset($data['lastname']))  ?
$data['lastname']     : null;
    }

    public function setInputFilter(InputFilterInterface $inputFilter)
    {
        throw new \Exception("Not used");
    }

    public function getInputFilter()
    {
        if (!$this->inputFilter) {
            $inputFilter = new InputFilter();
            $factory     = new InputFactory();

            $inputFilter->add($factory->createInput(array(
                'name'     => 'id',
                'required' => true,
                'filters'  => array(
                    array('name' => 'Int'),
                ),
            )));

            $inputFilter->add($factory->createInput(array(
                'name'     => 'username',
                'required' => true,
                'filters'  => array(
                    array('name' => 'StripTags'),
                    array('name' => 'StringTrim'),
                ),
                'validators' => array(
                    array(
                        'name'    => 'StringLength',
                        'options' => array(
                            'encoding' => 'UTF-8',
                            'min'      => 4,
                            'max'      => 100,
                        ),
                    ),
                ),
            )));

            $inputFilter->add($factory->createInput(array(
                'name'     => 'password',
                'required' => true,
                'filters'  => array(
                    array('name' => 'StripTags'),
                    array('name' => 'StringTrim'),
                ),
                'validators' => array(
                    array(
                        'name'    => 'StringLength',
                        'options' => array(
                            'encoding' => 'UTF-8',
                            'min'      => 6,
                            'max'      => 100,
                        ),
                    ),
                ),
```

```
                ),
            )));

            $inputFilter->add($factory->createInput(array(
                'name'     => 'firstname',
                'required' => false
            )));

            $inputFilter->add($factory->createInput(array(
                'name'     => 'lastname',
                'required' => false
            )));

            $this->inputFilter = $inputFilter;
        }

        return $this->inputFilter;
    }
}
```

UserTable.php

```
<?php
namespace Application\Model;

use Zend\Db\TableGateway\TableGateway;

class UserTable
{
    protected $tableGateway;

    public function __construct(TableGateway $tableGateway)
    {
        $this->tableGateway = $tableGateway;
    }

    public function fetchAll()
    {
        $resultSet = $this->tableGateway->select();
        return $resultSet;
    }

    public function getUser($id)
    {
        $id  = (int) $id;
        $rowset = $this->tableGateway->select(array('id' => $id));
        $row = $rowset->current();
        if (!$row) {
            throw new \Exception("Could not find row $id");
        }
        return $row;
    }

    public function getByUsername($username)
    {
        $rowset = $this->tableGateway->select(array('username' =>
$username));
        $row = $rowset->current();
        if (!$row) {
            return false;
        }

        return $row;
```

```
    }

    public function usernameAvailable($username)
    {
        $rowset = $this->tableGateway->select(array('username' =>
$username));
        $row = $rowset->current();
        if (!$row) {
            return true;
        }

        return false;
    }

    public function authUser($username, $hash)
    {
        $rowset = $this->tableGateway->select(array(
            'username' => $username,
            'password' => $hash
        ));
        $row = $rowset->current();
        if (!$row) {
            return true;
        }

        return false;
    }

    public function saveUser(User $user)
    {
        $data = array(
            'username'  => $user->username,
            'password'  => $user->password,
            'salt'      => $user->salt,
            'firstname' => $user->firstname,
            'lastname'  => $user->lastname
        );

        $id = (int)$user->id;
        if ($id == 0) {
            $this->tableGateway->insert($data);
        } else {
            if ($this->getUser($id)) {
                $this->tableGateway->update($data, array('id' => $id));
            } else {
                throw new \Exception('Form id does not exist');
            }
        }
    }

    public function deleteUser($id)
    {
        $this->tableGateway->delete(array('id' => $id));
    }
}
```

category/add.phtml

```
<div class="row-fluid">
    <div class="span12">
        <div class="row-fluid">
            <?php
            $form = $this->form;
```

```
            $form->setAttribute('action', $this->url('category_add'));
            $form->prepare();

            echo $this->form()->openTag($form);
            echo $this->formRow($form->get('name'));
            echo $this->formSubmit($form->get('submit'));
            echo $this->form()->closeTag();
            ?>
        </div>
    </div>
</div>
```

category/list.phtml

```
<p class="text-center">
    <a href="<?= $this->url('category_add') ?>" class="btn btn-
success">Create</a>
</p>
<div class="row-fluid">
    <div class="span12">
        <div class="row-fluid">
            <table class="table table-striped">
                <tbody>
                    <?php foreach ($categories as $category) {?>
                    <tr>
                        <td><?php echo $category->name; ?></td>
                        <td>
                            <a class="btn btn-small" href="<?= $this-
>url('category_remove', array(
                                'action'    => 'remove',
                                'id'        => $category->id
                                )) ?>">Remove</a>
                        </td>
                    </tr>
                    <?php } ?>
                </tbody>
            </table>
        </div>
    </div>
</div>
```

index/index.phtml

```
<div class="row-fluid">
    <div class="span12">
        <?php if ($this->success === true) { ?>
        <div class="alert alert-success">
            <p class="text-center">You have successfully registered</p>
        </div>
        <?php } ?>
        <div class="row-fluid">
            <div class="span6 well">
                <h3>Login</h3>
                <?php
                $form = $this->loginForm;
                $form->setAttribute('action', $this->url('login'));
                $form->prepare();

                echo $this->form()->openTag($form);
                echo $this->formRow($form->get('username'));
                echo $this->formRow($form->get('password'));
```

```
                echo $this->formSubmit($form->get('submit'));
                echo $this->form()->closeTag();
                ?>
            </div>
            <div class="span6 well">
                <h3>Registration</h3>
                <?php
                $form = $this->form;
                $form->setAttribute('action', $this->url('home'));
                $form->prepare();

                echo $this->form()->openTag($form);
                echo $this->formHidden($form->get('id'));
                echo $this->formRow($form->get('username'));
                echo $this->formRow($form->get('password'));
                echo $this->formRow($form->get('firstname'));
                echo $this->formRow($form->get('lastname'));
                echo $this->formSubmit($form->get('submit'));
                echo $this->form()->closeTag();
                ?>
            </div>
        </div>
    </div>
</div>
```

task/add.phtml

```
<div class="row-fluid">
    <div class="span12">
        <div class="row-fluid">
            <?php
            $form = $this->form;
            $form->setAttribute('action', $this->url('task_add'));
            $form->prepare();

            echo $this->form()->openTag($form);
            echo $this->formRow($form->get('title'));
            echo $this->formRow($form->get('description'));
            echo $this->formRow($form->get('progress'));
            echo $this->formRow($form->get('event_time'));
            echo $this->formRow($form->get('category_id'));
            echo $this->formRow($form->get('status_id'));
            echo $this->formSubmit($form->get('submit'));
            echo $this->form()->closeTag();
            ?>
        </div>
    </div>
</div>

<script>
    $(function(){
        $(".datepicker").datetimepicker({
            dateFormat: 'yy-mm-dd',
            separator: ' ',
            minDate: new Date('-1Hours')
        });
    });
</script>
```

task/index.phtml

```
<p class="text-center">
    <a href="<?= $this->url('task_add') ?>" class="btn btn-success">New
Task</a>
    <a href="<?= $this->url('category_add') ?>" class="btn btn-success">New
Category</a>
</p>
<div class="row-fluid">
    <div class="span12">
        <div class="row-fluid">
           <?php foreach ($categories as $category) {?>
                <h4><?php echo $category->name; ?></h4>
                <table class="table">
                <thead>
                    <tr>
                        <td>time</td>
                        <td>title</td>
                        <td>status</td>
                        <td>progress</td>
                        <td></td>
                    </tr>
                </thead>
                <?php foreach ($category->tasks as $task) { ?>
                    <tr
                <?php if ($task->status_id->active) { ?>class="info"<?php }
?>>
                        <td class="span2">
                            <p>
                                <?php if ($task->event_time) { ?>
                                    <span class="badge
                <?php if ($task->status_id->active) { ?>badge-info<?php }
?>">
                                        <?= $task->event_time ?>
                                    </span>
                                <?php } ?>
                            </p>
                        </td>
                        <td class="span3">
                            <p class="text-info">
                                <strong>
                                    <?= $task->title ?>
                                </strong>
                            </p>
                        </td>
                        <td class="span2">
                            <p>
                                <span class="badge
                <?php if ($task->status_id->active) { ?>badge-success<?php
} ?>">
                                    <?= $task->status_id->name ?>
                                </span>
                            </p>
                        </td>
                        <td class="span2">
                            <div class="progress progress-striped"
title="<?= (int)$task->progress.'%' ?>">
                                <div class="bar" style="width: <?=
(int)$task->progress.'%' ?>;"></div>
                            </div>
                        </td>
                        <td class="span3">
                            <a href="<?= $this->url('task_update',
array('id'    => $task->id)) ?>" class="btn btn-small">Update</a>
                            <a href="<?= $this->url('task_remove',
array('id'    => $task->id)) ?>" class="btn btn-small">Remove</a>
```

```
                        </td>
                    </tr>
                    <tr>
                        <td colspan="5" class="span12">
                            <p><?= $task->description ?></p>
                        </td>
                    </tr>
                <?php } ?>
            </table>
            <?php } ?>
        </div>
    </div>
</div>
```

task/update.phtml

```
<div class="row-fluid">
    <div class="span12">
        <div class="row-fluid">
            <?php
            $form = $this->form;
            $form->setAttribute('action', $this->url('task_update', array(
                'id' => $this->id
            )));
            $form->prepare();

            echo $this->form()->openTag($form);
            echo $this->formHidden($form->get('id'));
            echo $this->formRow($form->get('title'));
            echo $this->formRow($form->get('description'));
            echo $this->formRow($form->get('progress'));
            echo $this->formRow($form->get('event_time'));
            echo $this->formRow($form->get('category_id'));
            echo $this->formRow($form->get('status_id'));
            echo $this->formSubmit($form->get('submit'));
            echo $this->form()->closeTag();
            ?>
        </div>
    </div>
</div>

<script>
    $(function(){
        $(".datepicker").datetimepicker({
            dateFormat: 'yy-mm-dd',
            separator: ' ',
            minDate: new Date('-1Hours')
        });
    });
</script>
```

layout/loggedInLayout.phtml

```
<?php echo $this->doctype(); ?>

<html lang="en">
    <head>
        <meta charset="utf-8">
        <?php echo $this->headTitle('ToDo ZF2'); ?>
```

```
        <?php echo $this->headMeta()->appendName('viewport', 'width=device-
width, initial-scale=1.0') ?>

        <!-- Le styles -->
        <?php echo $this->headLink(array('rel' => 'shortcut icon', 'type'
=> 'image/vnd.microsoft.icon', 'href' => $this->basePath() .
'/img/favicon.ico'))
                        ->prependStylesheet($this->basePath() .
'/css/bootstrap-responsive.min.css')
                        ->prependStylesheet($this->basePath() .
'/css/style.css')
                        ->prependStylesheet($this->basePath() .
'/css/jquery-ui-1.10.3.custom.min.css')
                        ->prependStylesheet($this->basePath() .
'/css/bootstrap.min.css') ?>

        <!-- Scripts -->
        <?php echo $this->headScript()->prependFile($this->basePath() .
'/js/html5.js', 'text/javascript', array('conditional' => 'lt IE 9',))
                                    ->prependFile($this->basePath() .
'/js/bootstrap.min.js')
                                    ->prependFile($this->basePath() .
'/js/jquery.min.js')
                ?>

    </head>
    <body>
        <div class="navbar navbar-inverse navbar-fixed-top">
            <div class="navbar-inner">
                <div class="container">
                    <a class="btn btn-navbar" data-toggle="collapse" data-
target=".nav-collapse">
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                    </a>
                    <a class="brand" href="<?php echo $this->url('home')
?>">ToDo</a>
                    <div class="nav-collapse collapse">
                        <ul class="nav">
                            <li class="active"><a href="<?php echo $this-
>url('home') ?>">Home</a></li>
                            <li><a href="<?php echo $this-
>url('categories') ?>">Categories</a></li>
                            <li><a href="<?php echo $this->url('logout')
?>">Logout</a></li>
                        </ul>
                    </div><!--/.nav-collapse -->
                </div>
            </div>
        </div>
        <div class="container">
            <?php echo $this->content; ?>
            <div class="row-fluid">
                <div class="span9">
                    <address>
                        <strong>(c) 2013 Rīga</strong><br>
                        Latvijas Universitāte<br>
                    </address>
                </div>
                <div class="span3">
                    <address>
                        <strong>Edmunds Beinarovičs</strong><br>
```

```
                        <a
href="mailto:edmund@beinarovic.lv">edmund@beinarovic.lv</a><br>
                        <abbr title="Phone">P:</abbr> +371 267 198 47
                    </address>
                </div>
            </div>
        </div> <!-- /container -->
        <?php echo $this->inlineScript()
                    ->prependFile($this->basePath() . '/js/time-picker-
extension.js')
                    ->prependFile($this->basePath() . '/js/jquery-ui-
1.10.3.custom.min.js') ?>
    </body>
</html>
```

Module.php

```
<?php
/**
 * Zend Framework (http://framework.zend.com/)
 *
 * @link      http://github.com/zendframework/ZendSkeletonApplication for
the canonical source repository
 * @copyright Copyright (c) 2005-2013 Zend Technologies USA Inc.
(http://www.zend.com)
 * @license   http://framework.zend.com/license/new-bsd New BSD License
 */

namespace Application;

use Zend\Mvc\ModuleRouteListener;
use Zend\Mvc\MvcEvent;
use Zend\Db\TableGateway\TableGateway;
use Zend\Db\ResultSet\ResultSet;
use Application\Model\User;
use Application\Model\UserTable;
use Zend\ModuleManager\Feature\ServiceProviderInterface;
use Zend\Authentication\Adapter\DbTable as DbTableAuthAdapter;
use Zend\Authentication\AuthenticationService;
use Application\Model\Category;
use Application\Model\CategoryTable;
use Application\Model\Task;
use Application\Model\TaskTable;
use Application\Model\Status;
use Application\Model\StatusTable;

class Module implements ServiceProviderInterface
{
    // Moduļa uzstādījumu datne
    public function onBootstrap(MvcEvent $e)
    {
        $eventManager        = $e->getApplication()->getEventManager();
        $moduleRouteListener = new ModuleRouteListener();
        $moduleRouteListener->attach($eventManager);
    }

    public function getConfig()
    {
        return include __DIR__ . '/config/module.config.php';
    }

    public function getAutoloaderConfig()
    {
```

```
        return array(
            'Zend\Loader\StandardAutoloader' => array(
                'namespaces' => array(
                    __NAMESPACE__ => __DIR__ . '/src/' . __NAMESPACE__,
                ),
            ),
        );
    }

    public function getServiceConfig()
    {
        return array(
            'factories' => array(
                'Application\Model\UserTable' =>  function($sm) {
                    $tableGateway = $sm->get('UserTableGateway');
                    $table = new UserTable($tableGateway);
                    return $table;
                },
                'UserTableGateway' => function ($sm) {
                    $dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');
                    $resultSetPrototype = new ResultSet();
                    $resultSetPrototype->setArrayObjectPrototype(new
User());
                    return new TableGateway('user', $dbAdapter, null,
$resultSetPrototype);
                },
                'Application\Model\CategoryTable' =>  function($sm) {
                    $tableGateway = $sm->get('CategoryTableGateway');
                    $table = new CategoryTable($tableGateway);
                    return $table;
                },
                'CategoryTableGateway' => function ($sm) {
                    $dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');
                    $resultSetPrototype = new ResultSet();
                    $resultSetPrototype->setArrayObjectPrototype(new
Category());
                    return new TableGateway('category', $dbAdapter, null,
$resultSetPrototype);
                },
                'Application\Model\StatusTable' =>  function($sm) {
                    $tableGateway = $sm->get('StatusTableGateway');
                    $table = new StatusTable($tableGateway);
                    return $table;
                },
                'StatusTableGateway' => function ($sm) {
                    $dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');
                    $resultSetPrototype = new ResultSet();
                    $resultSetPrototype->setArrayObjectPrototype(new
Status());
                    return new TableGateway('status', $dbAdapter, null,
$resultSetPrototype);
                },
                'Application\Model\TaskTable' =>  function($sm) {
                    $tableGateway = $sm->get('TaskTableGateway');
                    $table = new TaskTable($tableGateway);
                    return $table;
                },
                'TaskTableGateway' => function ($sm) {
                    $dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');
                    $resultSetPrototype = new ResultSet();
                    $resultSetPrototype->setArrayObjectPrototype(new
Task());
                    return new TableGateway('task', $dbAdapter, null,
$resultSetPrototype);
```

```
                },
                'AuthService' => function($sm) {
                        //My assumption, you've alredy set dbAdapter
                        //and has users table with columns : user_name
and pass_word
                        //that password hashed with md5
                    $dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');
                    $dbTableAuthAdapter = new
DbTableAuthAdapter($dbAdapter,
                        'user',
                        'username',
                        'password',
                        "sha1(CONCAT(salt))"
                    );

                    $authService = new AuthenticationService();
                    $authService->setAdapter($dbTableAuthAdapter);
                        $authService->setStorage($sm-
>get('SanAuth\Model\MyAuthStorage'));

                    return $authService;
                },
            ),
        );
    }

}
```

Bakalaura darbs “PHP ietvari tīmekļa vietņu izstrādei” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Edmunds Beinarovičs

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Vadītājs: Dr. Dat., profesors Kārlis Čerāns 04.06.2013.

Recenzents: Dr. Dat., profesors Māris Vitiņš

Darbs iesniegts Datorikas fakultātē 2013. g. .....

Pieņēma .....

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

2013. g. .....

Komisijas sekretāris/e: .....