

УНИВЕРЗИТЕТ СИНГИДУНУМ
ДЕПАРТМАН ЗА ПОСЛЕДИПЛОМСКЕ СТУДИЈЕ

МАСТЕР ПРОГРАМ

САВРЕМЕНЕ ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

**ПРЕДЛОГ РЕШЕЊА ИНФОРМАЦИОНОГ СИСТЕМА
ЗА АНАЛИЗУ МЕДИЈА У МИНИСТАРСТВУ ОДБРАНЕ
БАЗИРАНОГ НА ZEND FRAMEWORK 2 ПЛАТФОРМИ**

- МАСТЕР РАД -

Ментор:
проф. др Младен Веиновић

Кандидат:
Благота Вуковић

Београд, 2016.

САДРЖАЈ

1	УВОД.....	4
2	МЕТОДОЛОГИЈА ИСТРАЖИВАЧКОГ РАДА.....	5
2.1	ПРЕДМЕТ РАДА.....	5
2.2	ЦИЉ ИСТРАЖИВАЊА.....	5
2.3	НАУЧНЕ МЕТОДЕ КОЈЕ СУ КОРИШЋЕНЕ.....	5
3	ИНФОРМАЦИОНИ СИСТЕМ.....	6
3.1	УВОД.....	6
3.2	СИСТЕМ.....	6
3.3	ИНФОРМАЦИОНИ СИСТЕМ.....	7
4	АНАЛИЗА МЕДИЈА.....	7
4.1	USE CASE ДИЈАГРАМ АНАЛИЗЕ МЕДИЈА.....	8
4.2	КОНЦЕПТУАЛНИ МОДЕЛ.....	9
5	ДИЈАГРАМИ СЕКВЕНЦИ СИСТЕМСКИХ ЗАХТЕВА.....	10
5.1	УНОС ПРИЛОГА.....	10
5.2	БРИСАЊЕ ПРИЛОГА.....	10
5.3	ИЗМЕНА ПРИЛОГА.....	11
5.4	ПРЕТРАГА ПРИЛОГА.....	11
5.5	ПРЕГЛЕД ПРИЛОГА.....	11
6	ZEND FRAMEWORK 2.....	12
6.1	УВОД.....	12
6.2	КАКО ZF2 РАДИ.....	12
6.3	ZF2 АРХИТЕКТУРА И MVC.....	13
6.4	ФОЛДЕРСКА СТРУКТУРА ZF2 АПЛИКАЦИЈА.....	14
7	СТРУКТУРА БАЗЕ ПОДАТАКА СИСТЕМА ЗА АНАЛИЗУ МЕДИЈА.....	15
7.1	ОСНОВНИ ПОЈМОВИ.....	15
7.2	ОПИС ТАБЕЛА.....	15
8	ОПИС МОДУЛА СА ПРИМЕРИМА КОДА.....	20
8.1	ZF2 МОДУЛИ.....	20
8.2	МОДУЛ ЗА РАД СА ПРИЛОЗИМА.....	21
8.2.1	<i>Редослед извршавања акција.....</i>	<i>22</i>
8.2.2	<i>Конфигурациони фајл.....</i>	<i>22</i>
8.2.3	<i>Контролер.....</i>	<i>22</i>
8.2.4	<i>Форма за унос прилога.....</i>	<i>23</i>
8.2.5	<i>Entity класа.....</i>	<i>23</i>
8.2.6	<i>Model класа.....</i>	<i>24</i>
8.3	МОДУЛ ЗА АУТЕНТИКАЦИЈУ.....	24
8.3.1	<i>Форма за пријављивање на систем.....</i>	<i>24</i>
8.3.2	<i>Role-Based Access Control (RBAC).....</i>	<i>25</i>
8.3.3	<i>Filter класа.....</i>	<i>25</i>
8.3.4	<i>Controller класа.....</i>	<i>26</i>
8.3.5	<i>Factory класе.....</i>	<i>26</i>
9	ТРАНСФЕР ПОДАТАКА ПУТЕМ ВЕБ СЕРИСА.....	26
10	ДОДАТНА УНАПРЕЂЕЊА СИСТЕМА.....	27
11	ЗАКЉУЧАК.....	29
12	СПИСАК СЛИКА.....	30

13	ЛИТЕРАТУРА.....	31
14	ПРИЛОЗИ	32

1 УВОД

Министарство одбране (МО) Републике Србије је ресор Владе Републике Србије задужен за одбрану државе од унутрашњих и спољашњих војних претњи, па је због тога добар део пословних процеса подређен одбрамбеним припремама и интерног је карактера, боље речено поверљивог.

Једна од организационих целина у Министарству одбране је Управа за односе са јавношћу (УОЈ), чија је основна улога да унапређује и развија облике модерне комуникације и информисања у функцији достизање циљева Министарства одбране и Војске Србије. Основни стратешки циљ је афирмација процеса реформе система одбране, професионализације војске и међународне сарадње, ради бољег разумевања и добијања подршке интерне и екстерне јавности за те процесе.

Управа је надлежна за: афирмацију система одбране у земљи и иностранству; остваривање информативних, промотивних и едукативних циљева кроз издавачку делатност, интернет, видео, филмску и телевизијску продукцију; пружање сервисних информација домаћим и страним јавним гласилима из области одбране.[1]

Дакле, Управа за односе са јавношћу, представља спону између војног система, који је по природи посла доста резервисан по питању изласка у јавност, са једне стране и грађанства, цивилних структура и медија са друге стране.

Један од пословних процеса који се реализују у Управи за односе са јавношћу је анализа медија из јавних гласила. У циљу оптимизације пословних процеса, део који се односи на прикупљање и обраду материјала за припрему дневног прес клипинга је outsource-ован, за што је ангажована специјализована агенција изван система одбране. Како се услуга обезбеђује путем јавних набавки на годишњем нивоу, методологија размене и трансфера података се разликује од агенције до агенције. У таквим условима јавља се проблем хетерогености система и ниске ефикасности у размени података.

Аутоматизација пословних процеса који се односе на анализу медија, у систему одбране датира још од 1993. године. У зависноти од развоја информационих технологија, развијани су и унапређивани информациони системи за ову намену. У овом раду предложено је решење Информационог система за анализу медија у Министарству одбране, базирано на Zend Framework 2 (ZF2) платформи, који представља један од модерних open-source објектно оријентисаних фрејмворка за развој веб апликација.

2 МЕТОДОЛОГИЈА ИСТРАЖИВАЧКОГ РАДА

2.1 Предмет рада

Предмет истраживања у овом раду је Информациони систем за анализу медија у Министарству одбране и његова имплементација заснована на Zend Framework 2 платформи. У раду ће бити описан поступак анализе медија у МО, уз употребу предложеног решења информационог система. Поред тога биће приказане, могућности употребе ZF2 open-source, објектно-оријентисаног фрејморка за развој веб апликација са конкретним примерима имплементације.

2.2 Циљ истраживања

Циљ истраживања је да се предложи решење ИС који ће да унапреди електронско пословање у МО које се односи на анализу медија и дневно извештавање по питању усмерености прилога из јавних гласила чија је тема Систем одбране у целини. Да би се решење реализовало, потребно је сагледати пословне процесе у Управи за односе са јавношћу МО које се односе се анализу медија, и препознати могућу примену савремених информационих технологија за израду веб апликације у виду ZF2 платформе.

2.3 Научне методе које су коришћене

У раду су примењене следеће научне методе:

- Аналитичке основне методе: метода анализе, метода апстракције, метода специјализације и дедукција
- Синтетичке основне методе: синтеза, конкретизација, генерализација и индукција
- Општенаучне методе: хипотетичко-дедуктивна метода, статистичка општенаучна метода, метода моделовања, упоредна метода
- Методе прикупљања података: анализа садржаја

3 ИНФОРМАЦИОНИ СИСТЕМ

3.1 Увод

У свакодневном разговору, не правимо увек јасну разлику између термина „податак“ и „информација“. Међутим, разлика између ова два појма је од виталног значаја за разумевање појма информационих система. Податак је низ чињеница, које су без значаја ако се посматрају засебно. Подаци могу бити у облику бројева када изражавамо нумеричка својства, у облику текста када изражавамо квалитативна својства, у облику слике и тако даље. Физички симболи којима се подаци бележе, региструју, морају бити погодни за чување пренос и обраду.

У свакодневном животу се срећемо са различитим проблемима. За решавање проблема некад постоји само једно решење, а некад их има више. Кад има више решења онда морамо донети одлуку које ћемо решење применити. Одлуку доносимо на основу једне или више чињеница које су нам познате у вези са датим проблемом. Ако само на основу једног податка доносимо одлуку тада тај податак представља информацију. Можемо још рећи да су подаци сировина из које се обрадом добијају информације. Или краће, „податак“ је **кодирана** „информација“, а „информација“ је **протумачени** податак

Циљ информационог система је да обради скуп чињеница на неки начин, тако да се као резултат добије корисна информација.

3.2 Систем

Систем (од латинског *systema*, узетог од грчког *σύστημα* *systema*) представља скуп елемената, реалних или апстрактних, представљених као целина где сваки елементи међусобно интерреагују су повезани са бар једним другим елементом и сви заједно служе заједничкој сврси.[4]

Систем има следеће собине:

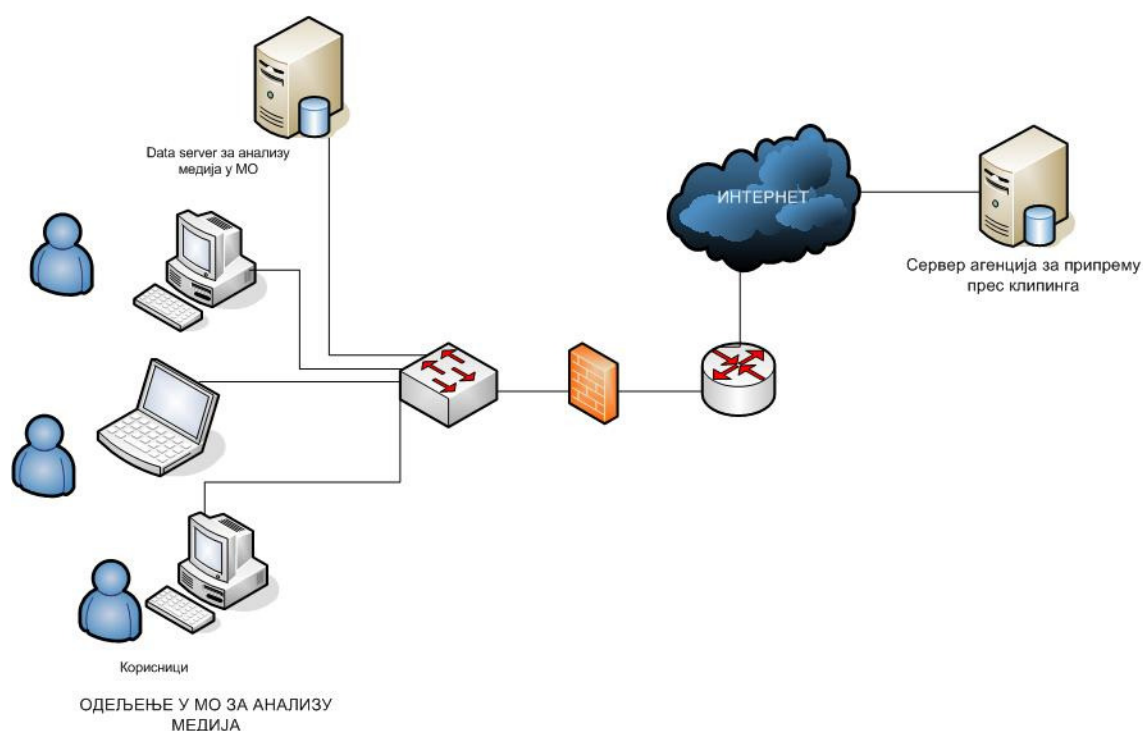
- Систем је организовани скуп компоненти са посебним везама између њих.
- Систем ради нешто, тј. представља модел понашања, јединствен за тај систем, или има специфичну сврху или циљ.
- Свака компонента утиче на одређени начин на понашање система као и на његово постојање. Ако се нека компонента уклони, промениће се и понашање система.

Не чини свака случајна колекција елемената систем. Потребно је да постоји одређена структура која има неки облик уређења, модел и сврху.

3.3 Информациони систем

Информациони систем је интегрисани скуп компоненти за сакупљање, снимање, чување, обраду и преношење информација. Рачунарски информациони систем је систем састављен од људи и рачунара који обрађују или интерпретирају информације. Појам се понекад користи у смислу да укаже само на софтвер који покреће рачунарске базе података или рачунарски систем.

Информациони систем који је описан у овом раду спада у групу **дистрибуираних информационих система** (ДИС) заснованих на клијент сервер архитектури. Дистрибуирани систем се састоји од више самосталних рачунара који комуницирају путем рачунарске мреже. Рачунари међусобно делују са намером да постигну заједнички циљ.

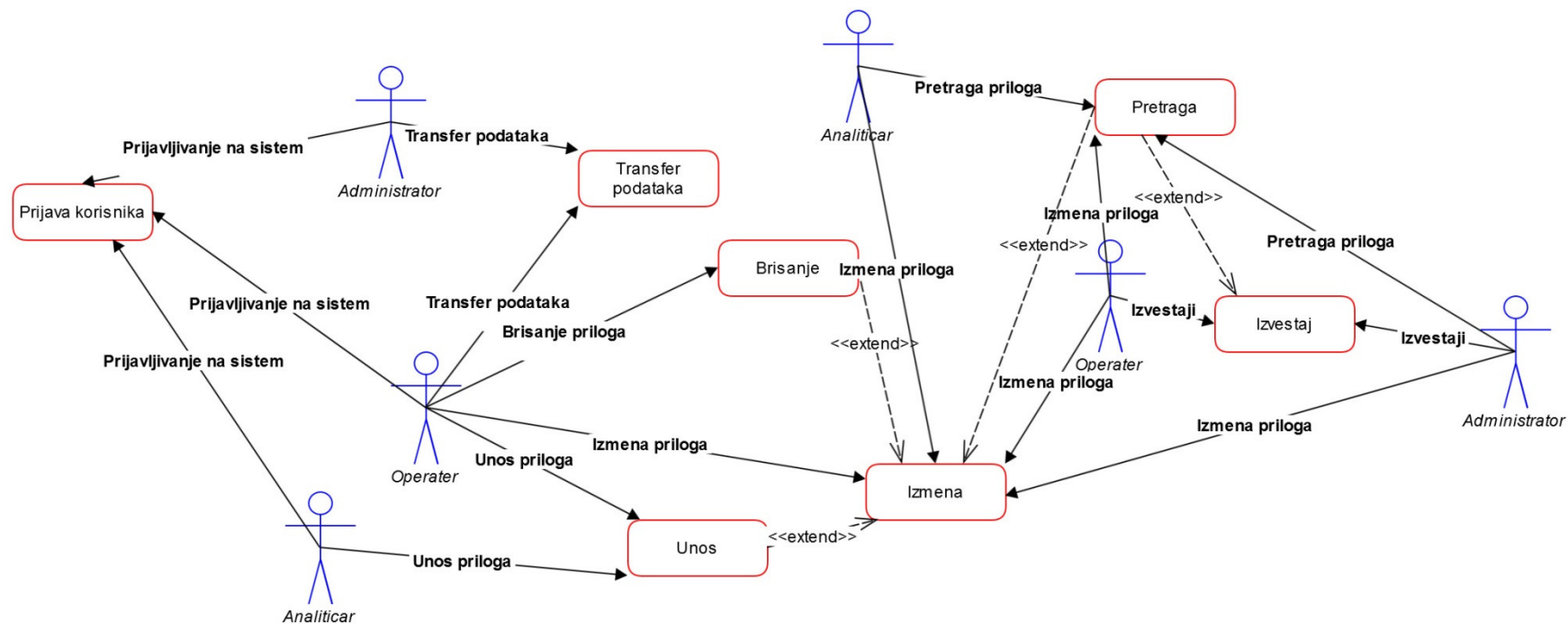


Слика 1: Физичка архитектура система за анализу медија

4 АНАЛИЗА МЕДИЈА

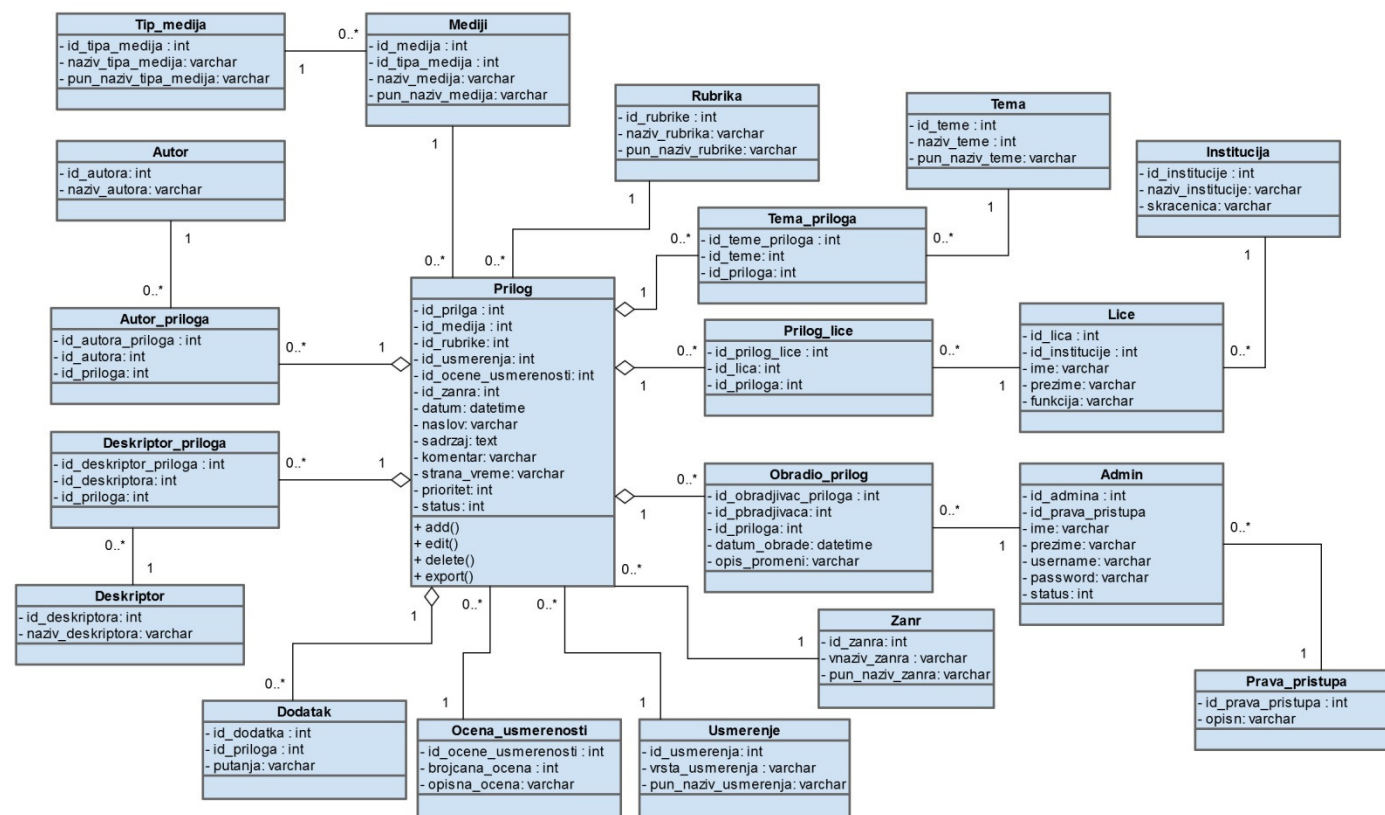
Анализу медија у организационом смислу обавља Одељење са n извршилаца различитог профила са одређеним радним задацима повезаних у основне функције. Обједињене функције чине процес анализе медија. [5]

4.1 Use case дијаграм анализе медија



Слика 2: Use case дијаграм анализе медија

4.2 Концептуални модел

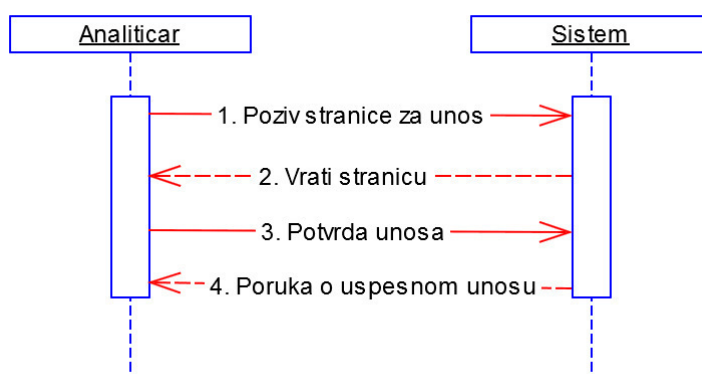


Слика 3: Концептуални модел Система за анализу медија

5 ДИЈАГРАМИ СЕКВЕНЦИ СИСТЕМСКИХ ЗАХТЕВА

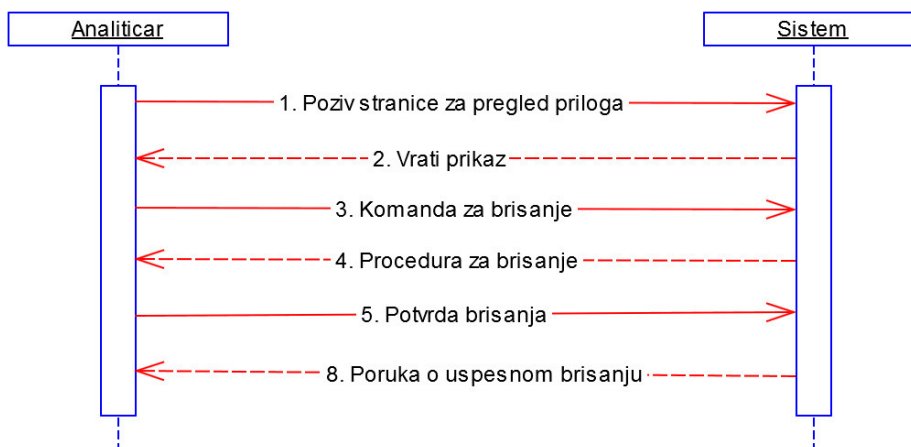
Дијаграм секвенци системских опарација приказује интеракцију корисника система и самог система, односно догађаје који се генеришу у тој интеракцији и који се односи на конкретан сценарио случаја коришћења. Догађај који иницира корисник система је управо побуда за позив системских операција. Дијаграм секвенци нам описује понашање система са циљем да добијемо одговоре на питање „шта систем ради“, не улазећи у проблематику како то ради.

5.1 Унос прилога



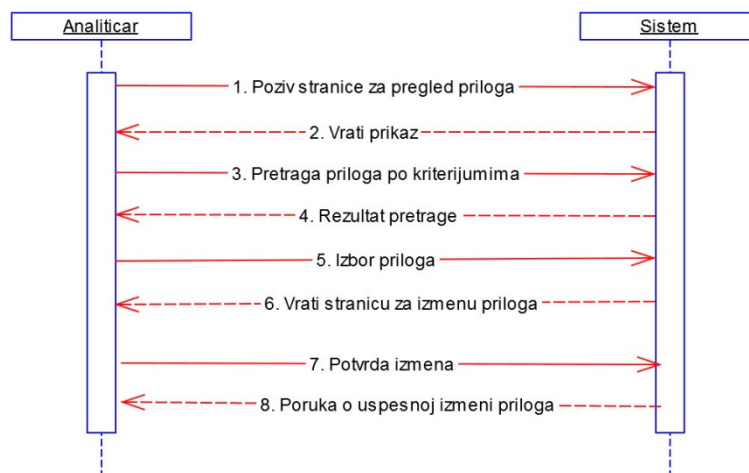
Слика 4: Унос прилога

5.2 Брисање прилога



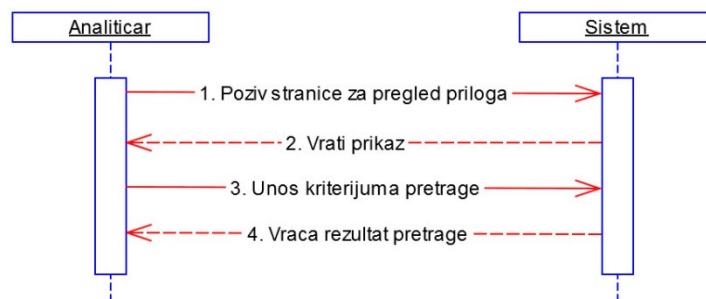
Слика 5: Брисање прилога

5.3 Измена прилога



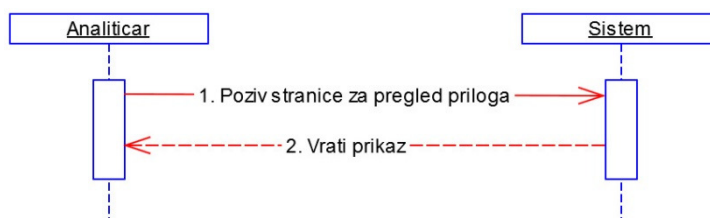
Слика 6: Измена прилога

5.4 Претрага прилога



Слика 7: Претрага прилога

5.5 Преглед прилога



Слика 8: Преглед прилога

6 ZEND FRAMEWORK 2

6.1 Увод

Zend Framework 2 (ZF2) је објектно-оријентисани open-source PHP фрејмворк за израду WEB апликација. ZF2 је унапређена верзија, тј. наставак развоја Zend Framework 1 (ZF1) пројекта. [6]

Основне карактеристике:

- Све компоненте су одрађене користећи објектно-оријентисани PHP
- Кориси Berkeley Software Distribution (BSD) лиценцу, дакле комплетан код је отвореног типа
- Захтева PHP 5.3.3 или напреднију верзију и користи предности PHP карактеристика
- „Use-at-will“ архитектура која омогућава минималну зависност међу компонентама
- Проширива Model View Controller (MVC) имплементација која подржава PHP layout-e
- Подржава више Система за управљање базама података (СУБП) као што су: MySQL, Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL, SQLite, MariaDB и Informix Dynamic Server
- Користи Git или GitHub за дељење кода
- Омогућава инсталирање само неопходних компоненети, а не цео фрејмворк
- Основни концепт је заснован на две кључне компоненте: Event Manager и Service Manager
- Израда модула који омогућавају систем електронске поште путем: mbox, Maildir, POP3 и IMAP
- Флексибилан систем кеширања меморије и фајл-система

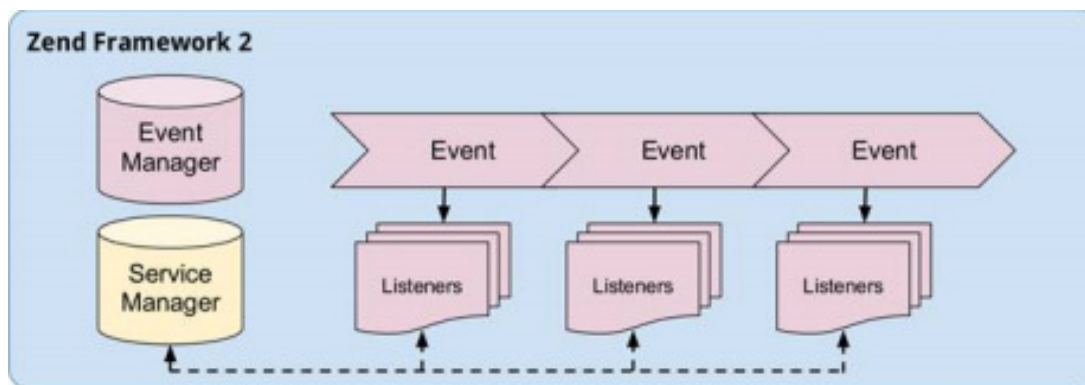
6.2 Како ZF2 ради

Основни концепт функционисања ZF2 се заснива на два кључна менаџера Event Manager и Service Manager, и listeners.

Event Manager служи да „испали“ **event** (акција која извршава неки део кода).

Listeners се придружују одређеном event-у и „ослушкује“ **triggers** (окидач) како би се извршила одређена акција.

Service Manager успоставља везу како би све компоненте фрејмворка биле доступне и складно функционисале.



Слика 9: Функционисање ZF2

6.3 ZF2 архитектура и MVC

MVC (Model-view-controller) је софтверска архитектура који одваја приказ информација од интеракције корисника са тим информацијама.

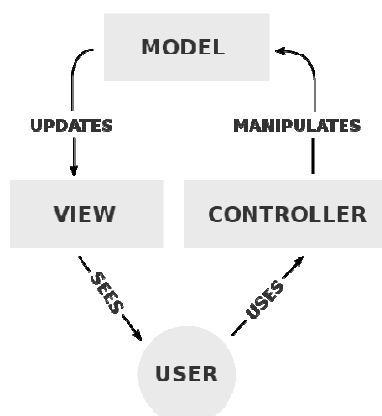
- слој пословне логике (Business logic layer)
- презентациони слој (Presentation layer)
- слој података (Data access layer)

MVC знатно утиче на организацију и читљивост програмског кода, па је постао стандард у писању модерних веб апликација.

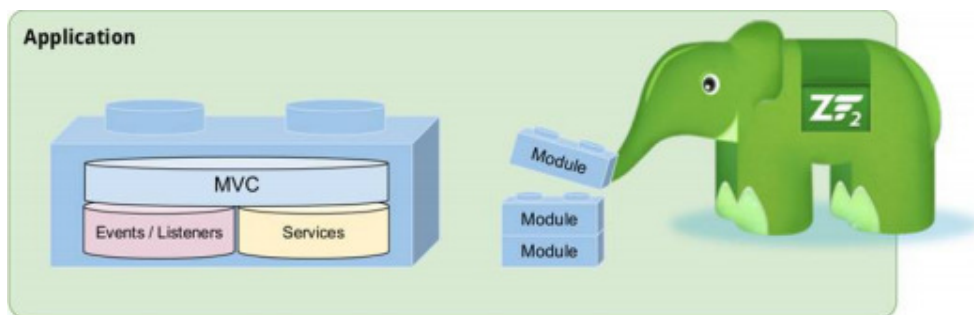
Model - имплементира слој пословне логике. Model је апстракција података тј. представља податке одређене апликације, управља понашањем података те реагује на захтеве за промену стања (обично из контролера).

View – приказује презентацију претходно моделованих података и задужен је за приказивање логике апликације.

Controller – управља корисничким захтевима те иницира одговор тако што позива на моделовање података.

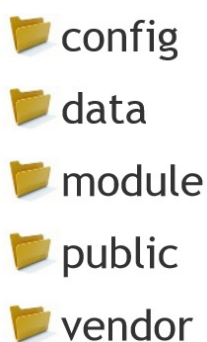


Слика 10: Повезаност компоненти у MVC архитектури

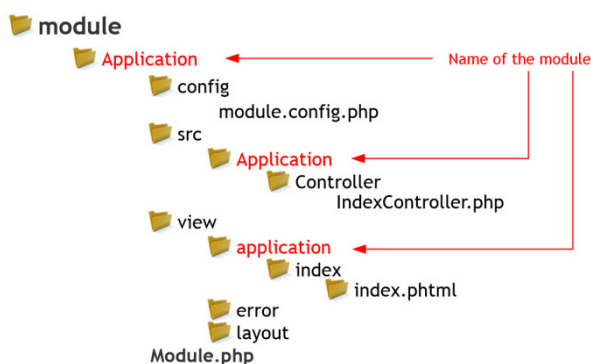


Слика 11: Корелација између ZF2 и MVC архитектуре

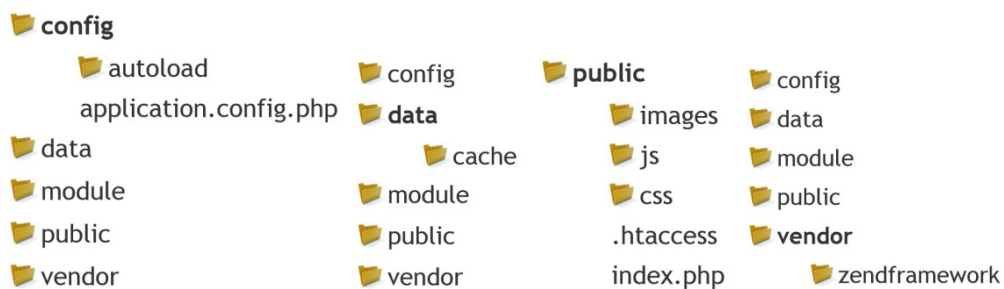
6.4 Фолдерска структура ZF2 апликација



Слика 12: Препоручена фолдерска структура ZF2 апликације



Слика 13: Структура фолдера **module/**



Слика 14. Структура фолдера **config/**, **data/**, **public/** и **vendor/**

Config фолдер – служи за смештање конфигурационих фајлова на нивоу апликације (фајлови за конекцију на базу података, конфигурациони фајлови за слање електронске поште и сл.).

Data фолдер – служи за смештање привремених и променљивих података. Погрешни подаци у овом фолдеру могу проузроковати лош рад апликације. Неки од фајлова који могу бити смештени на овој локацији су: сесијски фајлови, кеш фајлови, sqlite фајлови за базе података, логови и сл.

Module фолдер – служи за смештај програмских модула. Око 99% програмског кода које програмер напише, смешта се у овај фолдер.

Public фолдер – служи за смештање свих фајлова који трба да буду јавно доступни на нивоу апликације нпр. фотографије, текстуалне датотеке које треба да буду доступне за преузимање. То је у ствари *webroot* (коренски фолдер) наше апликације.

Vendor фолдер – служи за смештање *3rd party* модула (то су модули који нису наменски прављени за конкретну апликацију, али који доприносе функционисању појединих сегмената исте). Иницијални фолдер који се налази на овој локацији је **zendframework/** у коме се налазе сви фајлови фрејмворка.

7 СТРУКТУРА БАЗЕ ПОДАТАКА СИСТЕМА ЗА АНАЛИЗУ МЕДИЈА

7.1 Основни појмови

Као Систем за управљање базама података користи се *MySQL*. База података има 21 табелу. **Database engine** је *MyISAM*, **character set (charset)** је *utf8*, а **collation** је *utf8_unicode_ci*.

Database engine је софтверска компонента система за управљање базама података која се користи за унос, читање, измену и брисање података из базе података. То је тзв. *CRUD (Create, Read, Update and Delete)* механизам, који обухвата 4 основне функције за манипулацију са подацима у бази података.

Character set је скуп симбола и кодирања.

Collation је скуп правила за упоређивање карактера у *character set*-у

7.2 Опис табела

Прилог

Основна табела у коју се смештају подаци о прилозима је *am_prilog*. Њена структура изгледа овако.

``prilogid`` (поље је целобројног типа, аутоинкремент),

`medijid` (поље је целобројног типа),
`rubrikaid` (поље је целобројног типа),
`usmerenjeid` (поље је целобројног типа),
`ocena\_usmerenostiid` (поље је целобројног типа),
`zanrid` (поље је целобројног типа),
`datum` (поље ја датумског типа),
`naslov` (поље ја знаковног типа дужине 255 карактера),
`sadrzaj` (поље ја текстуалног типа),
`komentar` (поље ја знаковног типа дужине 255 карактера),
`strana\_vreme` (поље ја знаковног типа дужине 8 карактера),
`id` (поље ја знаковног типа дужине 50 карактера),
`prioritet` (поље је целобројног типа),
`statusp` (поље ја набројивог типа са вредностима 'обрађен',
'необрађен', 'у_току', 'обрисан'),

Примарни кључ је поље *prilogid*

ШИФАРНИЦИ

Аутор

Табела *am_autor* чува податке о ауторима новинских чланака, односно прилога из јавних гласила.

`autorid` (поље је целобројног типа, аутоинкремент),
`naziv\_autora` (поље ја знаковног типа дужине 50 карактера),
`ime` (поље ја знаковног типа дужине 50 карактера),
`osevo\_ime` (поље ја знаковног типа дужине 50 карактера),
`prezime` (поље ја знаковног типа дужине 50 карактера),
`inicijali` (поље ја знаковног типа дужине 5 карактера),
`statusa` (поље је целобројног типа),

Примарни кључ је поље *autorid*

Дескриптор

У табели *am_deskriptor* се чувају подаци о дескрипторима прилога. Дескриптор је реч или група речи која ближе описује прилог, и на тај начин га разврстава у одређене групе у оквиру неке теме.

`deskriptorid` (поље је целобројног типа, аутоинкремент),
`naziv\_deskriptora` (поље ја знаковног типа дужине 255 карактера),
`pun\_naziv\_deskriptora` (поље ја знаковног типа дужине 255 карактера),
`statusp` (поље је целобројног типа),

Примарни кључ је поље *deskriptorid*

Додатак

Додатак је фајл који иде уз прилог. То може да буде ПДФ или Word документ, аудио или видео запис. У табели *am_dodatak* се чувају подаци о додацима.

`dodatakid` (поље је целобројног типа, аутоинкремент),
`prilogid` (поље је целобројног типа),
`putanja` (поље ја знаковног типа дужине 255 карактера),
`statusd` (поље је целобројног типа),

Примарни кључ је поље *dodatakid*

Лице

Подаци о лицима која се помињу у прилогу смештају се у табели *am_lice*.

`liceid` (поље је целобројног типа, аутоинкремент),
`institucijaid` (поље је целобројног типа),

`ime` (поље ја знаковног типа дужине 50 карактера),
`prezime` (поље ја знаковног типа дужине 50 карактера),
`funkcije` (поље ја знаковног типа дужине 50 карактера),
`statusl` (поље је целобројног типа),

Примарни кључ је поље *liceid*

Институција

Подаци о институцијама којима припадају лица која се помињу у прилогу се чувају у табели *am_institucija*.

`institucijaid` (поље је целобројног типа, аутоинкремент),
`naziv\_institucije` (поље ја знаковног типа дужине 255 карактера),
`skracenice` (поље ја знаковног типа дужине 10 карактера),
`statusi` (поље је целобројног типа),

Примарни кључ је поље *institucijaid*

Медији

Подаци о медијима којима припадају прилози се чувају у табели *am_mediji*.

`medijid` (поље је целобројног типа, аутоинкремент),
`tip\_medijaid` (поље је целобројног типа),
`naziv\_medija` (поље ја знаковног типа дужине 255 карактера),
`pun\_naziv\_medija` (поље ја знаковног типа дужине 255 карактера),
`statusm` (поље је целобројног типа),

Примарни кључ је поље *medijid*

Оцена усмерености

Оцена усмерености је појам којим аналитичар даје оцену прилогу на основу контекста у коме је написан. Оцена усмерености може да буде: екстремно негативна, негативна, неутрална, позитвна и изразито позитивна. Подаци о оценама усмерености се чувају у табели *am_ocena_usmerenosti*.

`ocena\_usmerenostiid` (поље је целобројног типа, аутоинкремент),
`brojcana\_ocena` (поље је целобројног типа),
`opisna\_ocena` (поље ја знаковног типа дужине 50 карактера),
`statuso` (поље је целобројног типа),

Примарни кључ је поље *ocena_usmerenostiid*

Рубрика

Шифарник рубрика којима припадају прилози из неког медија је табела *am_rubrika*.

`rubrikaid` (поље је целобројног типа, аутоинкремент),
`naziv\_rubrike` (поље ја знаковног типа дужине 50 карактера),
`pun\_naziv\_rubrike` (поље ја знаковног типа дужине 50 карактера),
`statusr` (поље је целобројног типа),

Примарни кључ је поље *rubrikaid*

Тема

Шифарник тема којима припада прилог је табела *am_tema*.

`temaid` (поље је целобројног типа, аутоинкремент),
`naziv\_teme` (поље ја знаковног типа дужине 50 карактера),
`pun\_naziv\_teme` (поље ја знаковног типа дужине 50 карактера),
`statust` (поље је целобројног типа),

Примарни кључ је поље *temaid*

Тип медија

Типови медија могу бити: штампани, електронски и новинска агенција. Ти подаци се чувају у табели *am_tip_medija*.

```
`tip_medijaid` (поље је целобројног типа, аутоинкремент),  
`naziv_tipa_medija` (поље ја знаковног типа дужине 50 карактера),  
`statustm` (поље је целобројног типа),
```

Примарни кључ је поље *tip_medijaid*

Усмерење

Усмерење је описна оцена садржаја прилога, односно контекста у коме је садржај написан у оквиру неке теме. Усмерење може бити позитивно, неутрално и негативно. Подаци о усмерењима се чувају у табели *am_usmerenje*.

```
`usmerenjeid` (поље је целобројног типа, аутоинкремент),  
`vrsta_usmerenja` (поље ја знаковног типа дужине 50 карактера),  
`pun_naziv_usmerenja` (поље ја знаковног типа дужине 50 карактера),  
`statusu` (поље је целобројног типа),
```

Примарни кључ је поље *usmerenjeid*

Жанр

Шифарник жанрова се налази у табели *am_zanr*.

```
`zanrid` (поље је целобројног типа, аутоинкремент),  
`naziv_zanra` (поље ја знаковног типа дужине 50 карактера),  
`pun_naziv_zanra` (поље ја знаковног типа дужине 50 карактера),  
`statusz` (поље је целобројног типа),
```

Примарни кључ је поље *zanrid*

Корисници

Подаци о корисницима апликације се чувају у табели *sis_admin*.

```
`id` (поље је целобројног типа, аутоинкремент),  
`ime` (поље ја знаковног типа дужине 255 карактера),  
`prezime` (поље ја знаковног типа дужине 255 карактера),  
`username` (поље ја знаковног типа дужине 255 карактера),  
`password` (поље ја знаковног типа дужине 255 карактера),  
`statusk` (поље је целобројног типа),  
`korisnicka_grupa` (поље је целобројног типа),  
`datum_izmene` (поље је датумског типа),  
`datum_unosa` (поље је датумског типа),  
`adminid` (поље је целобројног типа),
```

Примарни кључ је поље *id*

Права приступа

Права приступа могу бити: администратор, оператер и аналитичар. Подаци се чувају у табели *am_prava_pristupa*.

```
`pravo_pristupaid` (поље је целобројног типа, аутоинкремент),  
`prava` (поље ја знаковног типа дужине 50 карактера),  
`opis` (поље ја знаковног типа дужине 50 карактера),  
`status` (поље је целобројног типа),
```

Примарни кључ је поље *pravo_pristupaid*

Системске поруке

Апликација омогућава слање порука међу корисницима апликације у виду обавештења и информација. Ти подаци се чувају у табели *sis_poruke*.

```
`id` (поље је целобројног типа, аутоинкремент),  
`konverzacija` (поље ја знаковног типа дужине 50 карактера),  
`posiljalacid` (поље је целобројног типа),  
`primalacid` (поље је целобројног типа),  
`sadrzaj` (поље ја текстуалног типа),  
`procitano` (поље је целобројног типа),  
`datum` (поље је датумског типа),
```

Примарни кључ је поље *id*

МЕЋУТАБЕЛЕ

Пошто кардиналност између појединих табела може да буде *више према више* било је неопходно увести међу табеле које решавају овај проблем.

Прилог - Аутор

Табела *am_autor_priloga*

```
`autor_prilogaid` (поље је целобројног типа, аутоинкремент),  
`prilogid` (поље је целобројног типа),  
`autorid` (поље је целобројног типа),
```

Примарни кључ је поље *autor_prilogaid*

Прилог - Дескриптор

Табела *am_deskriptor_priloga*

```
`deskriptor_prilogaid` (поље је целобројног типа, аутоинкремент),  
`deskriptorid` (поље је целобројног типа),  
`prilogid` (поље је целобројног типа),
```

Примарни кључ је поље *deskriptor_prilogaid*

Прилог - Корисници

Табела *am_obradio_prilog*

```
`obradjivac_prilogaid` (поље је целобројног типа, аутоинкремент),  
`obradjivacid` (поље је целобројног типа),  
`prilogid` (поље је целобројног типа),  
`datum_obrade` (поље је датумског типа),  
`opis_promene` (поље ја знаковног типа дужине 255 карактера),
```

Примарни кључ је поље *obradjivac_prilogaid*

Прилог - Лице

Табела *am_prilog_lice*

```
`prilog_liceid` (поље је целобројног типа, аутоинкремент),  
`liceid` (поље је целобројног типа),  
`prilogid` (поље је целобројног типа),
```

Примарни кључ је поље *prilog_liceid*

Прилог - Тема

Табела *am_tema_priloga*

```
`tema_prilogid` (поље је целобројног типа, аутоинкремент),
```

`temaid` (поље је целобројног типа),
`prilogid` (поље је целобројног типа),
Примарни кључ је поље *тема_prilogid*

8 ОПИС МОДУЛА СА ПРИМЕРИМА КОДА

8.1 ZF2 модули

ZF2 модул је скуп фолдера и фајлова који имплементирају одређену функционалност система. Да би један модул могао да функционише у оквиру апликације потребно је да има следеће (као пример ће бити приказан модул за манипулацију са прилозима у оквиру система):

- Фајл *Module.php*
- *Namespace* са истим називом као и назив модуле (у нашем случају то је **Prilog**)
- И да модул буде пријављен на нивоу апликације у фајлу *application.config.php*

Детаљна структура модуле Prilog изгледа на следећи начин:

Prilog/

Module.php

autoload_classmap.php

autoload_function.php

autoload_register.php

config/

module.config.php

public/

images/

css/

js/

src/

Prilog/

Controller/

PrilogController.php

Form/

PrilogForm.php

PrilogFilter.php

Entity

Prilog.php

Model

PrilogRepository.php

test/

phpunit.xml

bootstrap.php

Prilog/

SampleTest.php

view/
 prilog/
 prilog/
 index.phtml
 add.phtml
 edit.phtml

8.2 Модул за рад са прилозима

Модул за рад са прилозима је одговорана за приказ, додавање, измену, брисање прилога као и остале подрадње које се одвијају приликом обраде података које се односе на овај део система.

Информациони систем за анализу медија у МО и ВС

ПРИЛОЗИ

Прилози

Страна < 1 > од 2 | Прикажи 10 записа | Приказано укупно 12 записа

Изабери корисни Изабери радњу.

✓ Потврди

ИД	Наслов	Датум	Медији	Рубрика	Усмерење	Оцена усмерености	Жанр	Страна/Време	Статус
1	Војска Србије и Министарство одбране	12.08.2015.	Политика	Политика	Негативно	Негативно	Информативни	1	обрађен
2	Промоција потпоручника	11.08.2015.	Блиц	Дневник 2	Позитивно	Позитивно	Интерпретативни	15	необрађен
3	Унапређења и систему одбране	11.08.2015.	Вечерње новости	Друштво-хроника	Неутрално	Негативно	Фотопублицистички	5	обрисан
4	Војска помаже угроженом становништву	08.08.2015.	Политика	Београдска хроника	Неутрално	Позитивно	Фотопублицистички	5	у току
5	Српски мировњаци на нивоу задатка	05.08.2015.	Блиц	Дневник 2	Негативно	Изразит поз.	Информативни	6	обрађен
6	Војни лекари лече становништво у неприступачним селима	03.08.2015.	Вечерње новости	Друштво-хроника	Неутрално	Позитивно	Фотопублицистички	9	необрађен
7	Припреме за војну параду и јеку	03.08.2015.	РТС1	Дневник 2	Негативно	Екстремно нег.	Интеракцијски	3	необрађен
8	Додељене војне спомен-медаље	09.08.2015.	Студио Б	Дневник 2	Негативно	Позитивно	Информативни	9	у току
9	Проглашени најбољи студенти на Војној академији	04.08.2015.	РТС2	Друштво-хроника	Позитивно	Неутрално	Интеракцијски	6	обрађен
10	Падобранци оправдали рејтинг	05.08.2015.	РТС2	Политика	Негативно	Неутрално	Информативни	7	обрисан

Страна < 1 > од 2 | Прикажи 10 записа | Приказано укупно 12 записа

2015 © Информациони систем за анализу медија у МО и ВС

Слика 15. Изглед табеле за приказ прилога

8.2.1 Редослед извршавања акција

Да би се у веб прегледачу приказала било која страница апликације, неопходно је да тај захтев буде обрађен од стране контролера, који мора бити пријављен у конфигурационом фајлу.

8.2.2 Конфигурациони фајл

Као пример имамо страницу за приказ прилога која се отвара прозивањем УРЛ-а <http://analizamedija/prilog/> кроз прегледач. Апликација тада проверава да ли је контролер који разрешава упит регистрован у конфигурационом фајлу модула и да ли постоји исправна рута за ту акцију. Део кода из конфигурационог фајла `module.config.php` приказан је у прилогу 1.

Конфигурациони фајл за модул садржи вишедимензиони асоцијативни низ са комбинацијом елемената кључ => вредност.

Кључ **controllers** има вредности путање до контролера који разрешавају захтеве. Да би контролер био у стању да разреши захтев, он мора да буде регистрован у конфигурационом фајлу, односно мора да постоји пар (кључ, вредност), у оквиру кључа **controllers**

Кључ **router** као вредност назове рута који могу да се разреши неким контролером. Те руте практично представљају УРЛ-ове који се прозивају у прегледачу, након чега треба да се изврши одређена акција

Кључ **view_manager** има вредности који су путање до фајлова који представљају сегмент презентационог дела апликације.

Асоцијативни низ може да садржи још елемената којима се дефинишу разне функционалности система. Примери кључева су **service_manager**, **translator**, **caches** итд.

8.2.3 Контролер

За разрешавање сценарија у нашем премеру (приказ преилога) одговоран је контролер **PrilogController.php**. Део кода који реализује акцију приказа прилога приказа је у прилогу 2.

У коду је приказана класа **PrilogController** која дефинише методе за рад са прилозима. Методе могу да имају прилагођена имена. Поред тога методе које реализују неку акцију имају заједнички суфикс – *Action*, док се први део назива методе разликује сходно функционалности коју та метода треба да оствари (*index*, *add*, *edit*, *delete*...). Такве методе се обично извршавају када се кроз веб прегледач прозове УРЛ, чија рута мора бити дефинисана у конфигурационом фајлу модула.

Метода `indexAction()` је одговорна за приказ табеле прилога, `addAction()` дефинише поступак за додавања прилога, `deleteAction()` реализује функционалност брисања прилога.

8.2.4 Форма за унос прилога

До форме за унос прилога се долази прозивањем УРЛ-а у веб прегледачу <http://analizamedija/prilog/add>, тада се активира контролер који помоћу методе *editAction()* разрешава захтев. Та метода практично прави инстанцу класе *PrilogForm*, која има дефинисане елементе форме које треба попунити подацима за унос у базу података. Та форма се исписује у фајлу *edit.phtml* који представља View за контролер *PrilogController*.

Слика 16. Пример изгледа форме за унос прилога

Део кода који извршава приказ форме за унос прилога приказан је у прилогу 3.

8.2.5 Entity класа

У директоријуму *Entity* налазе се фајлови са *Entity* класама које обично имају назив као и табела из базе података на коју се односе. У нашем примеру то је класа *Prilog*. Класа има атрибуте који представљају ентитете из табеле, и потребно је имплементира две методе, *exchangeArray()* и *getArrayCopy()*.

Метода *exchangeArray()* служи за упис података у табелу тако што врши декомпозицију објекта који јој се прослеђује на ентитете табеле.

Метода *getArrayCopy()* омогућава дохватање података из табеле и њихово смештање у објекат, који служи за обраду у апликативној логици.

Поред ове две методе, на основу атрибута се генеришу и гетери и сетери. Гетери служе да се путем њих обезбеди приступ атрибуту, а сетери служе за доделу вредности атрибуту. Део кода ентитетске класе *Prilog* је приказан у прилогу 4.

8.2.6 Model класа

У директоријуму *Model* налазе се класе које служе за манипулацију са базом података. То је посредни елемент између *Controller-a* и *View-a* код MVC архитектуре. Називи класа се састоје из назива табеле и суфикса *Repository*, ради лакше организације фајлова. Део кода класе *PrilogRepository()* приказан је у прилогу 5.

8.3 Модул за аутентикацију

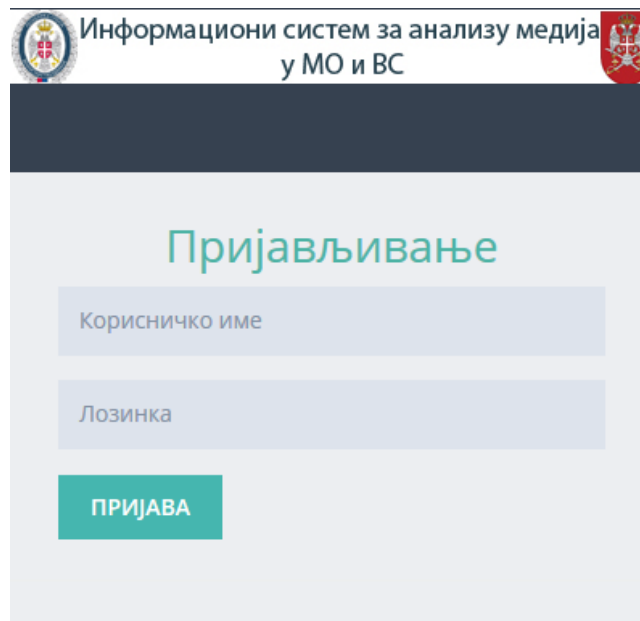
Модул за аутентикацију је врло битан део система јер обезбеђује да само овлашћени корисник може да користи ресурсе система. Пријављивање на систем корисник реализује уношењем корисничког имена и лозике. Тиме се активира механизам ауторизације и аутентикације. *Аутентикација* представља процес утврђивања идентитета неког корисника, што у праски значи да тај корисник мора бити уписан у базу корисника система. *Ауторизација* представља механизам којим се утврђује да ли пријављени корисник може да приступи одређеном ресурсу система. То се постиже подешавањем одговарајућих права приступа.

8.3.1 Форма за пријављивање на систем

Приликом покретања апликације, врши се провера конфигурационог фајла модула. Део кода је приказан у прилогу 6.

Кориснику се отвара форма за пријављивање на систем. Корисник уноси корисничко име у форми налога за електронску пошту (и-мејл) и лозинку. Део кода који генерише *HTML* елемента форме приказан је у прилогу 7.

На основу овог кода генерише се форма за пријављивање приказана на слици 15.



Слика 17. Форма за пријављивање на систем

8.3.2 Role-Based Access Control (RBAC)

RBAC је део ZF2 и представља модул који разрешава ауторизацију на нивоу апликације. Прилагођени модул који проширује класу *Rbac*, неопходан за рад апликације је *CheckRbac*. Приликом пријављивања на систем, а након исправног проласка механизма за аутентикацију и ауторизацију, кориснички подаци се уписују у сесију. На тај начин за сваку страницу може да се провери да ли пријављени корисник има право приступа. *Module.php* у оквиру модуле *CheckRbac* приказан је у прилогу 8.

Метода *onBootstrap()* се извршава сваки пут када се покрене модул. Та метода „ослушкује“ понашање апликације и у моменту када је то неопходно позива методу *onDispatch()* у којој се врши инстанцирање објекта класе *Rbac* која проширује класу *Rbac* која је у оквиру ZF2. Део кода класе *Rbac* је приказан у прилогу 9.

Постоје 3 корисничке групе на нивоу система. Непријављени корисника - *guest*, оператер – *user* и технички администратор - *admin*. Права приступа су дефинисана системом наслеђивања тј. оператер наслеђује права приступа непријављеног корисника, а администратор наслеђује права приступа оператера.

Модул *CheckRbac* мора бити пријављен на нивоу апликације, као и било који други модул. За то је задужен конфигурациони фајл апликације, а његов део у коме су дефинисани модули приказан је у прилогу 10.

8.3.3 Filter класа

У процесу аутентикације и ауторизације приликом пријављивања на систем, активира се модуле *CheckRbac* који проширује класу *ZendRbac*, системску класу у оквиру *ZF2*. *Module.php* фајл тог модула разрешава унешене параметре од стране корисника и након обраде враћа поруку или ради редирекцију на неку од страница апликације.

Почетна страница апликације приказана на слици 16, приказује форму за пријављивање на систем која припада модулу *Sistem*. Приликом уноса података и кликом на дугме *Пријава* активира се филтер који је задужен да онемогући унос неодговарајућих података. Класа *LoginFilter* је приказана у прилогу 11.

8.3.4 Controller класа

Controller класа разрешава упите које се односе на модул коме припада. Упити се разрешавају тако што се покрећу акције у оквиру контролера. Акција која се активира када се отвори страница за пријаву на систем је приказана у прилогу 12.

8.3.5 Factory класе

Factory класе имплементирају *FactoryInterface* и служи за управљање ентитетима за које се формирају. У модулу *Sistem* имамо три *factory* класе:

1. AdminTableFactory, прилог 13
2. LoginFormFactory, прилог 14
3. LoginControllerFactory, прилог 15

9 ТРАНСФЕР ПОДАТАКА ПУТЕМ ВЕБ СЕРИСА

Идеја је да се поступак трансфера податка, од стране агенције која ради прес клипинг за потребе МО, аутоматизује. У ту сврху је као оптимално решење изабран трансфер путем веб сервиса.

Веб сервис представља услугу коју један електронски уређај нуди другом електронском уређају комуницирајући међусобно путем **ВЕБ-а** (World Wide Web). Код веб сервиса, комуникациона технологија је прилагођена за комуникацију између две машине и за то се користе за машину читљиви формати података, најчешће **JSON** и **XML**. [7]

Протокол који се најчешће користи код веб сервиса је **SOAP** (Simple Object Access Protocol). Он је намењен за размену структурираних података у имплементацији веб сервиса у рачунарској мрежи. SOAP користи XML Information Set за формат поруке и ослања се на протоколе апликативног слоја OSI (Open Systems Interconnection) модела, најчешће HTTP (Hypertext Transfer Protocol) или SMTP (Simple Mail Transfer Protocol).

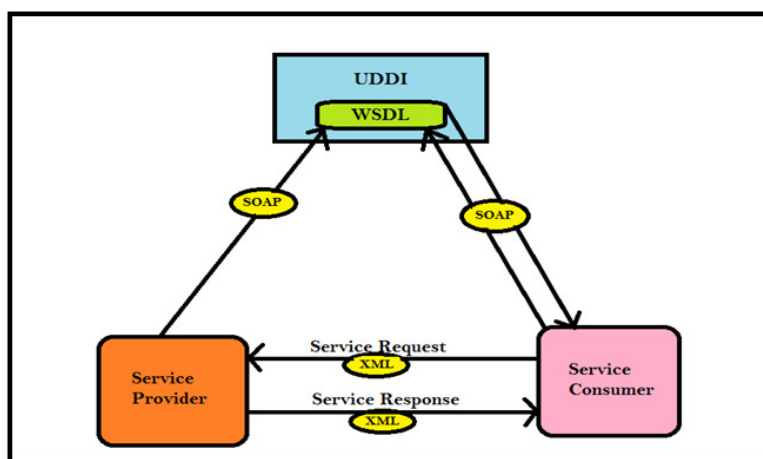
Један од стилова архитектуре у развоју веб сервиса је **REST** (Representational state transfer) који омогућава брже перформансе, поузданост и скалабилност сервиса.

То омогућава **Web API** (Application programming interface) који представља низ процедура и функција које се користе на клијентској страни комуникације. API омогућава једноставнију употребу ВЕБ сервиса.

Директоријум који се зове **UDDI** (Universal Description, Discovery and Integration) дефинише који ће системски софтвер бити контактиран за неки тип податка.

Да би се тачно знало какви подаци треба да доставе клијенту потребно је да се сервис опише, односно да се тачно прецизира који подаци се достављају, у ком формату и сл. У ту сврху се користи **WSDL** (Web Services Description Language). То је дакле фајл у коме су описана правила комуникације у сервису.

У циљу несметаног рада апликације, потребно је да прес клипинг агенција на својој страни реализује веб сервис, и да доставља скупове података у дефинисаним временским интервалима информационом сиситему за анализу медија.



Слика 18. Приказ трансфера података путем веб сервиса

10 ДОДАТНА УНАПРЕЂЕЊА СИСТЕМА

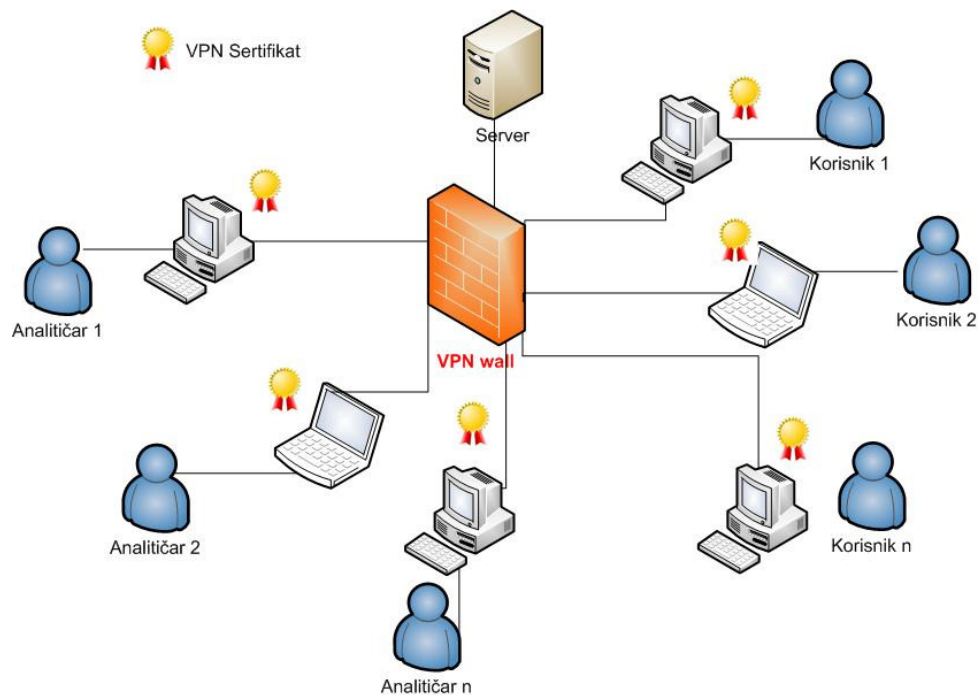
Једно од унапређења система је оно које се односи на аспект заштите. Идеја је да се поред апликативног нивоа заштите, који се односи на процес аутентикације и ауторизације путем корисничког имена и лозинке, уведе и мрежни ниво заштите. Један од начина да се то реализује је путем **ВПН** (Virtual Private Network).[8]

Виртуелна приватна мрежа је приватна комуникациона мрежа која се користи за комуникацију у оквиру јавне мреже. Транспорт ВПН пакета података одвија се преко јавне мреже (нпр. интернет) коришћењем стандардних комуникационих протокола. ВПН омогућава корисницима на раздвојеним локацијама да преко јавне мреже одржавају заштићену комуникацију.

ВПН је тунел кроз који се одвија заштићена комуникација између сервера и клијента. ВПН сервер се обично инсталира на уређају на коме се налази ресурс који треба да се заштити. Клијентска страна треба да има инсталирану клијентску апликацију ради конекције на ВПН сервер.

Конекција на ВПН сервер се одвија користећи два елемента. Један је оно што физички **имамо** код себе, а то је ВПН сертификат. Он може да буде инсталиран на фајл систем у виду датотеке или да буде уписан на „смарт“ каритци. Други елемент је оно што **знамо**, а то је лозинка која одговара сертификату.

Дакле, сваки приступ апликацији би морао бити остварен кроз ВПН конекцију, што значи да сваки корисник мора имати свој персонализовани ВПН сертификат за приступ.



Слика 19. Архитектура система са ВПН мрежом

11 ЗАКЉУЧАК

Сагледавањем пословних процеса који се односе на припрему прес клипинга у Министарству одбране, аутор рада је имао прилику да се упозна са постојећим софтверским решењем које се користи за ту намену, да уочи његове недостатке и да предложи унапређења система у складу са иновацијама на пољу веб технологија.

За ту намену узет је Zend Framework 2 као основа за развој веб апликације. У питању је модуларно софтверско решење које омогућава реализацију великог броја пословних процеса. На овај начин, софтверско решење је зановљено и прилагођено актуелним трендовима развоја веб апликација. Оваквим приступом олакшан је процес одржавања апликације јер морају да се поштују правила писања кода и креирања модула.

Употреба веб сервиса је још једно унапређење система. Таквим решењем повећава се аутоматизација процеса, јер је код имплементације веб сервиса суштина да се омогући комуникација на ниво систем-систем. Тиме се неутралишу грешке изазване људским фактором.

Са аспекта заштите, новина је могућност инсталације система кроз ВПН мрежу. На тај начин се омогућава приступ апликацији само корисницима који имају ВПН сертификат, што значи да критични ресурси имају већи степен заштите.

Имајући у виду значај медија у модерном времену, а посебно оних који спадају у домен електронских, овакав информациони систем може да допринесе у ефикасности обављања послова везаних за анализу медија. Нарочито треба истаћи чињницу да се ти послови одвијају у једном врло сложеном систему, који је непрекидно под лупом медија и друштва у целини у смислу одговорности и реализације послова за које је надлежан.

12 СПИСАК СЛИКА

РБ.	Назив слике
Слика 1	Физичка архитектура система за анализу медија
Слика 2	Use case дијаграм анализе медија
Слика 3	Концептуални модел Система за анализу медија
Слика 4	Унос прилога
Слика 5	Брисање прилога
Слика 6	Измена прилога
Слика 7	Претрага прилога
Слика 8	Преглед прилога
Слика 9	Функционисање ZF2
Слика 10	Повезаност компоненти у MVC архитектури
Слика 11	Корелација између ZF2 и MVC архитектуре
Слика 12	Препоручена фолдерска структура ZF2 апликације
Слика 13	Структура фолдера <i>module/</i>
Слика 14	Структура фолдера <i>config/, data/, public/ u vendor/</i>
Слика 15	Изглед табеле за приказ прилога
Слика 16	Пример изгледа форме за унос прилога
Слика 17	Форма за пријављивање на систем
Слика 18	Приказ трансфера података путем веб сервиса
Слика 19	Архитектура система са ВПН мрежом

13 ЛИТЕРАТУРА

1. www.mod.gov.rs[23.04.2016]
2. Примена Model Driven Arhicetkture у развоју система за електронско пословање система одбране, Зоран Бобар, ОТЕХ 2009
3. UML osnove objektnog modeliranja, prof. dr Alempije Veljović
4. <https://sr.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC> [07.07.2016]
5. https://en.wikipedia.org/wiki/Media_monitoring_service[03.05.2016]
6. <https://framework.zend.com/manual/2.4/en/index.html>[07.07.2016]
7. https://en.wikipedia.org/wiki/Web_service[18.07.2016]
8. https://en.wikipedia.org/wiki/Virtual_private_network[18.07.2016]
9. Using Zend Framewor 2, Oleg Kratsov, 2013
10. Zend Framework 2.0 by Example, Krishna Shasankar V, 2013
11. Web Development with Zend Framework 2, Concepts, Techniques and Practical Solutions, Michael Romer, 2013

14 ПРИЛОЗИ

Прилог 1:

```
<?php
return array(
    'controllers' => array(
        'invokables' => array(
            'Prilog\Controller\Prilog' =>
            'Prilog\Controller\PrilogController',
        ),
    ),
    'router' => array(
        'routes' => array(
            'prilog' => array(
                'type' => 'Literal',
                'options' => array(
                    'route' => '/prilog',
                    'defaults' => array(
                        '__NAMESPACE__' => 'Prilog\Controller',
                        'controller' => 'Prilog',
                        'action' => 'index',
                    ),
                ),
            ),
            'may_terminate' => true,
            'child_routes' => array(
                'add' => array(
                    'type' => 'Segment',
                    'options' => array(
                        'route' => '/add',
                        'defaults' => array(
                            'action' => 'add',
                        ),
                    ),
                ),
                'edit' => array(
                    'type' => 'Segment',
                    'options' => array(
                        'route' => '/edit[:id]',
                        'constraints' => array(
                            'id' => '[a-zA-Z0-9_-]*',
                        ),
                        'defaults' => array(
                            'action' => 'edit',
                        ),
                    ),
                ),
                'delete' => array(
                    'type' => 'Segment',
                    'options' => array(
                        'route' => '/delete[:id]',
                        'constraints' => array(
                            'id' => '[a-zA-Z0-9_-]*',
                        ),
                        'defaults' => array(
                            'action' => 'delete',
                        ),
                    ),
                ),
            ),
        ),
    ),
);
```

```

    ),
    'ajax' => array(
        'type' => 'Segment',
        'options' => array(
            'route' => '/ajax',
            'constraints' => array(
                'id' => '[a-zA-Z0-9_-]*',
            ),
            'defaults' => array(
                'action' => 'ajax',
            ),
        ),
    ),
    'akcija' => array(
        'type' => 'Segment',
        'options' => array(
            'route' => '[:action]',
            'constraints' => array(
                'action' => '[a-zA-Z][a-zA-Z0-9_-]*',
            ),
            'defaults' => array(
                'controller' => 'Prilog',
            ),
        ),
    ),
),
.....
...остале руте.....
.....
),
),
'view_manager' => array(
    'template_path_stack' => array(
        'Prilog' => __DIR__ . '/../view',
    ),
    'template_map' => array(
        'layout-prilog' => __DIR__ . '/../view/layout/layout-
prilog.phtml',
    ),
    'strategies' => array(
        'ViewJsonStrategy',
    ), ), );

```

Прилог 2:

```
<?php
namespace Prilog\Controller;

use Zend\Mvc\Controller\AbstractActionController;
use Prilog\Entity\Prilog;
use Zend\View\Model\JsonModel;
use Zend\Validator\File\Size;
use Zend\File\Transfer\Adapter\Http;
use Zend\Validator\File\Extension;
use Zend\EventManager\EventManager;

class PrilogController extends AbstractActionController
{
    public function indexAction()
    {
        $this->layout('layout-prilog');
        $repo = $this->serviceLocator->get('PrilogRepository');
        $dbAdapter = $this->serviceLocator->get('zend-db-adapter');
        return array(
            'dbAdapter' => $dbAdapter
        );
    }

    public function addAction()
    {
        $this->layout('layout-form');
        $form = $this->getServiceLocator()->get('prilog_form');
        $request = $this->getRequest();

        $authService = $this->getServiceLocator()->get('auth-
service');
        $identity = $authService->getIdentity();
        $userId = $identity->id;

        if ($request->isPost())
        {
            $nonFile = $request->getPost()->toArray();
            $File = $this->params()->fromFiles('slika');
            $foto = $File['name'];
            $data = array_merge(
                $nonFile,
                array('slika'=> $foto)
            );
            $form->setData($data);
            if ($form->isValid())
            {
                if ($foto == '')
                {
                    $this->createPrilog($form->getData());
                    $this->generateThumbnail($data['slika']); //generise se thumbnail
                    return $this->redirect()->toRoute('prilog');
                }
                else
                {
                    $adapter = new Http();
                    //max velicina fajla je 1MB
                }
            }
        }
    }
}
```

```

        $size_validator = new Size(array('max'=>'1MB'));
        //validacija za extenziju, mogu da se uploaduju
samo .jpg i .png i gif
        $extension_validator = new
Extension(array('extension'=>array('jpg','png','gif')));
        $adapter-
>setValidators(array($extension_validator, $size_validator), $foto);
        if (!$adapter->isValid())
        {
            $dataError = $adapter->getMessages();
            $error = array();
            foreach($dataError as $key=>$row)
            {
                $error[] = $row;
            }
            $form->setMessages(array('slika'=>$error ));
        }
        else
        {
            $adapter->setDestination($this-
>getFileUploadLocation());
            $foto = time().'_'.$File['name'];
            $adapter->addFilter('File\Rename',
                array('target' => $adapter-
>getDestination().'/'.$foto,
                    'overwrite' => true));

            if ($adapter->receive($File['name']))
            {
                $file = $adapter-
>getFilter('File\Rename')->getFile();

                $data = array_merge(
                    $nonFile,
                    array('slika'=> $foto)
                );
                $form->setData($data);

                if ($form->isValid())
                {
                    $this->createPrilog($form->getData());
                    $this-
>generateThumbnail($data['slika']);
                    return $this->redirect()-
>toRoute('prilog');
                }
            }
        }
    }
}

return array('form' => $form, 'currentUser' => $userId);
}

public function deleteAction()
{
    $prilogRepo = $this->getServiceLocator()-
>get('PrilogRepository');
    $prilogRepo->deletePrilog($this->params()->fromRoute('id'));
}

```

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

Прилог 3:

```
<?php
namespace Prilog\Form;

use Zend\Form\Form;
use Prilog\Model\AutorRepository;
use Prilog\Model\DeskriptorRepository;
use Prilog\Model\TemaRepository;
use Prilog\Model\TipMedijaRepository;
use Prilog\Model\ZanrRepository;

class PrilogForm extends Form
{
    protected $autorRepo;
    protected $deskriptorRepo;
    protected $temaRepo;
    protected $tipMedijaRepo;
    protected $zanrRepo;

    public function __construct(AutorRepository $autorRepo,
                                DeskriptorRepository $deskriptorRepo,
                                TemaRepository $temaRepo,
                                TipMedijaRepository $tipMedijaRepo,
                                ZanrRepository $zanrRepo){
        $this->setAutor($autorRepo);
        $this->setDeskriptor($deskriptorRepo);
        $this->setTema($temaRepo);
        $this->setTipMedija($tipMedijaRepo);
        $this->setZanr($zanrRepo);

        parent::__construct('Artikal');
        $this->setAttribute('method', 'post');
        $this->setAttribute('enctype', 'multipart/form-data');
        $this->setAttribute('class', 'form-horizontal');

        $this->add(array(
            'name' => 'id',
            'attributes' => array(
                'type' => 'hidden',
            ),
        ));

        $this->add(array(
            'name' => 'datum',
            'attributes' => array(
                'type' => 'hidden',
            ),
        ));

        $this->add(array(
            'name' => 'admin_id',
            'attributes' => array(
                'type' => 'hidden',
            ),
        ));
    }
}
```

```

$this->add(array(
    'name' => 'naslov',
    'attributes' => array(
        'type' => 'text',
        'class' => 'form-control input-large',
        'placeholder' => 'Наслов',
    ),
    'options' => array(
        'label' => 'Наслов',
    ),
));

$this->add(array(
    'type' => 'Zend\Form\Element\Select',
    'name' => 'autor',
    'attributes' => array(
        'class' => 'form-control input-large',
    ),
    'options' => array(
        'label' => 'Аутор',
        'value_options' => $this->getAutori(),
        'empty_option' => 'Изаберите...',
    ),
));

$this->add(array(
    'type' => 'Zend\Form\Element\Select',
    'name' => 'deskriptor',
    'attributes' => array(
        'class' => 'form-control input-large',
    ),
    'options' => array(
        'label' => 'Дескриптор',
        'value_options' => $this->getDeskriptori(),
        'empty_option' => 'Изаберите...',
    ),
));

$this->add(array(
    'type' => 'Zend\Form\Element\Select',
    'name' => 'tema',
    'attributes' => array(
        'class' => 'form-control input-large',
    ),
    'options' => array(
        'label' => 'Тема',
        'value_options' => $this->getTeme(),
        'empty_option' => 'Изаберите...',
    ),
));

$this->add(array(
    'type' => 'Zend\Form\Element\Select',
    'name' => 'tip_medija',
    'attributes' => array(
        'class' => 'form-control input-large',
    ),
    'options' => array(
        'label' => 'Тип медија',
    ),
));

```

```

        'value_options' => $this->getTipoviMedija(),
        'empty_option' => 'Изаберите...',
    ),
));

$this->add(array(
    'type' => 'Zend\Form\Element\Select',
    'name' => 'zanr',
    'attributes' => array(
        'class' => 'form-control input-large',
    ),
    'options' => array(
        'label' => 'Жанр',
        'value_options' => $this->getZanrovi(),
        'empty_option' => 'Изаберите...',
    ),
));

$this->add(array(
    'name' => 'slika',
    'attributes' => array(
        'type' => 'file',
    ),
    'options' => array(
        'label' => 'Слика',
    ),
));

$this->add(array(
    'type' => 'Zend\Form\Element\Radio',
    'name' => 'status',
    'attributes' => array(
        'class' => 'radio',
    ),
    'options' => array(
        'label' => 'Статус',
        'value_options' => array(
            'obradjen' => 'Обрађен',
            'neobradjen' => 'Необрађен',
            'obrisan' => 'Обрисан',
        ),
    ),
));

$this->add(array(
    'name' => 'snimi',
    'attributes' => array(
        'type' => 'submit',
        'class' => 'btn blue',
    ),
));

$this->add(array(
    'name' => 'cancel',
    'attributes' => array(
        'type' => 'button',
        'class' => 'btn default',
    ),
));

```

```

        'onclick' => 'window.history.back();'
    ),
));
}
.....
.....остале методе.....
.....
}

```

Прилог 4:

```
<?php
namespace Prilog\Entity;

class Prilog
{
    protected $id_priloga;
    protected $id_medija;
    protected $id_rubrike;
    protected $id_usmerenja;
    protected $id_ocene_usmerenosti;
    protected $id_zanra;
    protected $datum;
    protected $naslov;
    protected $sadrzaj;
    protected $komentar;
    protected $strana_vreme;
    protected $id;
    protected $prioritet;
    protected $status;

    public function exchangeArray($data)
    {
        $this->id_priloga = (isset($data['id_priloga'])) ?
$data['id_priloga'] : null;
        $this->id_medija = (isset($data['id_medija'])) ?
$data['id_medija'] : null;
        $this->id_rubrike = (isset($data['id_rubrike'])) ?
$data['id_rubrike'] : null;
        $this->id_usmerenja = (isset($data['id_usmerenja'])) ?
$data['id_usmerenja'] : null;
        $this->id_ocene_usmerenosti =
(isset($data['id_ocene_usmerenosti'])) ? $data['id_ocene_usmerenosti']
: null;
        $this->id_zanra = (isset($data['id_zanra'])) ? $data['id_zanra'] :
null;
        $this->datum = (isset($data['datum'])) ? $data['datum'] : null;
        $this->naslov = (isset($data['naslov'])) ? $data['naslov'] : null;
        $this->sadrzaj = (isset($data['sadrzaj'])) ? $data['sadrzaj'] :
null;
        $this->komentar = (isset($data['komentar'])) ? $data['komentar'] :
null;
        $this->strana_vreme = (isset($data['strana_vreme'])) ?
$data['strana_vreme'] : null;
        $this->id = (isset($data['id'])) ? $data['id'] : null;
        $this->prioritet = (isset($data['prioritet'])) ?
$data['prioritet'] : null;
        $this->status = (isset($data['status'])) ? $data['status'] : null;
    }

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

```

```
        return $this->id_priloga;
    }

    .....
    ...остали гетери и сетери...
    .....
}
```

Прилог 5:

```
<?php
namespace Prilog\Model;

use Zend\Db\Adapter\Adapter;
use Zend\Db\ResultSet\HydratingResultSet;
use Zend\Db\Sql\Expression;
use Zend\Stdlib\Hydrator\Reflection as ReflectionHydrator;
use Zend\Db\TableGateway\TableGateway;
use Prilog\Entity\Prilog;
use Zend\Db\Sql>Select;

class PrilogRepository
{
    protected $tableGateway;
    protected $hydrator;

    function __construct(Adapter $adapter)
    {
        $this->hydrator = new HydratingResultSet(
            new ReflectionHydrator(),
            new Prilog()
        );

        $this->tableGateway = new TableGateway('am_prilog', $adapter,
null, $this->hydrator);
    }

    public function savePrilog(Prilog $prilog)
    {
        $datum = strtotime($prilog->getDatum());

        $data = array(
            'id_priloga' => $prilog->getId_priloga(),
            'id_medija' => $prilog->getId_medija(),
            'id_rubrike' => $prilog->getId_rubrike(),
            'id_usmerenja' => $prilog->getId_usmerenja(),
            'id_ocene_usmerenja' => $prilog->getId_ocene_usmerenja(),
            'id_zanra' => $prilog->getId_zanra(),
            'datum' => new
\Zend\Db\Sql\Expression("FROM_UNIXTIME(".$datum.")"),
            'naslov' => $prilog->getNaslov(),
            'sadrzaj' => $prilog->getSadrzaj(),
            'komentar' => $prilog->getKomentar(),
            'strana_vreme' => $prilog->getStrana_vreme(),
            'id' => $prilog->getId(),
            'prioritet' => $prilog->getPrioritet(),
            'status' => $prilog->getStatus()
        );

        $id = (int)$prilog->getId_priloga();
        if ($id == 0) {
            $this->tableGateway->insert($data);
        } else {
            if ($this->getPrilog($id)) {
```

```

        $this->tableGateway->update($data,
array('rob_artikal.id' => $id));
    } else {
        throw new \Exception('Prilog ne postoji');
    }
}

public function getPrilog($id)
{
    $id = (int)$id;

    $rowset = $this->tableGateway-
>select(array('am_prilog.id_priloga' => $id));
    $row = $rowset->current();
    if (!$row) {
        throw new \Exception("Nema zapisa ciji je ID $id");
    }
    return $row;
}

.....
...остале методе....
.....
}

```

Прилог 6:

```
<?php
return array(
    'controllers' => array(
        'invokables' => array(
            'Sistem\Controller\Admin' =>
'Sistem\Controller\AdminController',
        ),
        'aliases' => array(
            'login-index' => 'login-factory',
            'system-admin' => 'Sistem\Controller\Admin',
        ),
        'factories' => array(
            'login-factory' =>
'Sistem\Service\LoginControllerFactory',
        )
    ),
    'router' => array(
        'routes' => array(
            'system-admin' => array(
                'type' => 'Literal',
                'options' => array(
                    'route' => '/system-admin',
                    'defaults' => array(
                        'controller' => 'system-admin',
                        'action' => 'index',
                    )
                )
            ),
            'may_terminate' => true,
            'child_routes' => array(
                'add' => array(
                    'type' => 'Segment',
                    'options' => array(
                        'route' => '/add',
                        'defaults' => array(
                            'action' => 'add',
                        )
                    )
                ),
                'edit' => array(
                    'type' => 'Segment',
                    'options' => array(
                        'route' => '/edit[:id]',
                        'constraints' => array(
                            'id' => '[a-zA-Z0-9_-]*',
                        )
                    ),
                    'defaults' => array(
                        'action' => 'edit',
                    )
                ),
                'delete' => array(
                    'type' => 'Segment',
                    'options' => array(
                        'route' => '/delete[:id]',
                        'constraints' => array(
                            'id' => '[a-zA-Z0-9_-]*',
                        )
                    )
                )
            )
        )
    )
);
```

```

        ),
        'defaults' => array(
            'action' => 'delete',
        ),
    ),
    ),
    'profileaccount' => array(
        'type' => 'Segment',
        'options' => array(
            'route' => '/profileaccount',
            'defaults' => array(
                'action' => 'profileaccount',
            ),
        ),
    ),
    ),
    'changepassword' => array(
        'type' => 'Segment',
        'options' => array(
            'route' => '/changepassword',
            'defaults' => array(
                'action' => 'changepassword',
            ),
        ),
    ),
    ),
    ),
    ),
    ),
    'service_manager' => array(
        'aliases' => array(
            'login-auth-service' => 'application-auth-service',
        ),
        'invokables' => array(
            'login-filter' => 'Sistem\Form\LoginFilter',
        ),
        'factories' => array(
            'admin-table' => 'Sistem\Service\AdminTableFactory',
            'users-table' => 'Sistem\Service\UsersTableFactory',
            'login-form' => 'Sistem\Service\LoginFormFactory',
        )
    ),
), ), );

```

Прилог 7:

```
<?php
namespace Sistem\Form;
use Zend\Form\Form as ZendForm;

class LoginForm extends ZendForm
{
    public function prepareElements()
    {
        $this->add(array(
            'name' => 'username',
            'options' => array(
                'label' => 'Username',
                'label_attributes' => array(
                    'class' => 'control-label visible-ie8 visible-
ie9'
                ),
            ),
            'attributes' => array(
                'type' => 'email',
                'class' => 'form-control form-control-solid
placeholder-no-fixr',
                'placeholder' => 'Корисничко име',
                'autocomplete' => true,
            ),
        ));

        $this->add(array(
            'name' => 'password',
            'options' => array(
                'label' => 'Password',
                'label_attributes' => array(
                    'class' => 'control-label visible-ie8 visible-ie9'
                ),
            ),
            'attributes' => array(
                'type' => 'password',
                'class' => 'form-control form-control-solid
placeholder-no-fixr',
                'placeholder' => 'Лозинка',
                'autocomplete' => false,
            ),
        ));

        $this->add(array(
            'name' => 'submit',
            'attributes' => array(
                'type' => 'submit',
                'value' => 'Пријава',
                'class' => 'btn btn-success uppercase',
            ),
        ));
    }
}
```

Прилог 8:

```
<?php
namespace CheckRbac;

use Zend\Mvc\MvcEvent;
use CheckRbac\Model\Rbac;

class Module
{
    public function onBootstrap(MvcEvent $e)
    {
        $eventManager = $e->getApplication()->getEventManager();
        $eventManager->attach(MvcEvent::EVENT_DISPATCH, array($this,
        'onDispatch'), 100);
    }

    public function onDispatch(MvcEvent $e)
    {
        // get routing info
        $matches = $e->getRouteMatch();
        $permission = $matches->getParam('controller') . '.' .
        $matches->getParam('action');
        // get services
        $sm = $e->getApplication()->getServiceManager();
        $authService = $sm->get('checkrbac-auth-service');
        $rbac = $sm->get('checkrbac-rbac');
        // retrieve identity
        $name = 'guest';
        $role = 'guest';
        if ($authService->hasIdentity()) {
            $identity = $authService->getIdentity();
            $name = $identity->ime . " " . $identity->prezime;
            $role = Rbac::getRoleFromStatusCode($identity->status);
        }
        // assign name and role to the layout view model
        $layoutView = $e->getViewModel();
        $layoutView->setVariable('realName', $name);
        $layoutView->setVariable('realRole', $role);

        if (!$rbac->hasRole($role) && $rbac->isGranted($role,
        $permission)) {
            $response = $e->getResponse();
            $headers = $response->getHeaders();
            $headers->clearHeaders();
            $headers->addHeaderLine('Location: /access-denied');
            $response->setStatusCode('302');
            $response->setHeaders($headers);
            return $response;
        }
    }

    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__,
            ),
        ),
    );
}

```

Прилог 9:

```
<?php
namespace CheckRbac\Model;

use Zend\Permissions\Rbac\Rbac as ZendRbac;
use Zend\Permissions\Rbac\Role;

class Rbac extends ZendRbac
{
    protected static $roles = array(0 => 'guest', 1 => 'user', 2 =>
    'admin');
    public static function getRoleFromStatusCode($statusCode)
    {
        return (isset(self::$roles[$statusCode])) ?
        self::$roles[$statusCode] : self::$roles[0];
    }
    public function __construct()
    {
        $guest = new Role('guest');
        $user = new Role('user');
        $admin = new Role('admin');

        $guest->addPermission('Application\Controller\Index.accessDenied')
        ->addPermission('login-index.index')
        ->addPermission('login-index.invalid')
        ->addPermission('login-index.confirm')
        ->addPermission('login-index.logout');

        $user->addPermission('Prilog\Controller\Prilog.index')
        ->addPermission('Prilog\Controller\Prilog.add')
        ->addPermission('Prilog\Controller\Prilog.edit')
        ->addPermission('Prilog\Controller\Prilog.delete')
        ->addPermission('Prilog\Controller\Prilog.ajax')
        ->addPermission('SanRestful\Controller\SampleClient.index')
        ->addPermission('SanRestful\Controller\SampleRestful.index');

        $admin->addPermission('sistem-admin.index')
        ->addPermission('sistem-admin.add')
        ->addPermission('sistem-admin.edit')
        ->addPermission('sistem-admin.delete')

        $user->addChild($guest);
        $admin->addChild($user);

        // add roles with inheritance
        $this->addRole($guest)
            ->addRole($user)
            ->addRole($admin); }}
```

Прилог 10:

```
<?php
return array(
    'modules' => array(
        'Prilog',
        'Sistem',
        'CheckRbac',
    ),
    .....
    ....остали елементи конфигурационог фајла....
    .....
);
```

Прилог 11:

```
<?php
namespace Sistem\Form;
use Zend\InputFilter\InputFilter;
use Zend\InputFilter\Input;
use Zend\Validator;
class LoginFilter extends InputFilter
{
    public function prepareFilters()
    {
        $username = new Input('username');
        $username->getFilterChain()
            ->attachByName('StripTags')
            ->attachByName('StringTrim');
        $username->getValidatorChain()
            ->addValidator(new Validator\EmailAddress());
        $username->setRequired(TRUE);

        $password = new Input('password');
        $password->getFilterChain()
            ->attachByName('StripTags')
            ->attachByName('StringTrim');
        $password->setRequired(TRUE);

        $this->add($username)
            ->add($password);
    }
}
```

Прилог 12:

```
public function indexAction()
{
    $this->layout('layout-login');
    $message = '';
    $data = '';
    $request = $this->getRequest();
    if ($request->isPost()) {
        $this->loginForm->setData($request->getPost());
        $valid = $this->loginForm->isValid();
        $data = $this->loginForm->getData();
        if (!$valid)
        {
            $this->logInvalidAttempt($data['username']);
            if ($this->setInvalidLoginCounter() > $this->maxInvalidAttempts) {
                $this->setInvalidLoginCounter(TRUE);
            }
            return $this->redirect()->toRoute('login-invalid');
        }
    }
    else
    {
        if ($data['new'] == 1) {
            $message .= $this->addUser($data);
        }
        else {
            $authService = $this->getAuthService();
            $authAdapter = $authService->getAdapter();
            $authAdapter->setIdentity($data['username']);
            $authAdapter->setCredential(md5($data['password']));
            $attempt = $authAdapter->authenticate();
            if ($attempt->isValid()) {
                $adminRepo = $this->getServiceLocator()->get('AdminRepository');
                $userRow = $authAdapter->getResultRowObject();
                $authStorage = $authService->getStorage();
                $authStorage->write($userRow);
                return $this->redirect()->toRoute('home');
            }
            else {
                $this->logInvalidAttempt($attempt->getIdentity());
                if ($this->setInvalidLoginCounter() > $this->maxInvalidAttempts)
                {
                    $this->setInvalidLoginCounter(TRUE); // reset counter
                }
                return $this->redirect()->toRoute('login-invalid');
            }
            else {
                $message = '<div class="alert alert-danger"><button class="close" data-close="alert"></button>Неуспешно пријављивање</div>'.implode('<br />', $attempt->getMessages()).'</div>';
            }
        }
    }
    return new ViewModel(array('loginForm' => $this->loginForm, 'message' => $message));
}
```

Прилог 13:

```
<?php
namespace Sistem\Service;

use Sistem\Model\AdminTable;
use Zend\ServiceManager\FactoryInterface;
use Zend\ServiceManager\ServiceLocatorInterface;

class AdminTableFactory implements FactoryInterface
{
    public function createService(ServiceLocatorInterface $services)
    {
        $adapter = $services->get('zend-db-adapter');
        return new AdminTable($adapter);
    }
}
```

Прилог 14:

```
<?php
namespace Sistem\Service;

use Sistem\Form\LoginForm;
use Zend\ServiceManager\FactoryInterface;
use Zend\ServiceManager\ServiceLocatorInterface;

class LoginFormFactory implements FactoryInterface
{
    public function createService(ServiceLocatorInterface $services)
    {
        $filter = $services->get('login-filter');
        $form = new LoginForm();
        $form->prepareElements();
        $filter->prepareFilters();
        $form->setInputFilter($filter);
        return $form;
    }
}
```

Прилог 15:

```
<?php
namespace Sistem\Service;

use Sistem\Controller\LoginController;
use Zend\ServiceManager\FactoryInterface;
use Zend\ServiceManager\ServiceLocatorInterface;
use Zend\Session\Container;

class LoginControllerFactory implements FactoryInterface
{
    public function createService(ServiceLocatorInterface $services)
    {
        $allServices = $services->getServiceLocator();
        $controller = new LoginController();
        $controller->setAdminTable($allServices->get('admin-table'));
        $controller->setLoginForm($allServices->get('login-form'));
        $controller->setSessionContainer(new Container('login'));
        return $controller;
    }
}
```