



Univerzitet Singidunum

**Departman za poslediplomske studije
i međunarodnu saradnju**

Savremene informacione tehnologije

Master studije

Aplikacija eMentor

- master rad -

Mentor:

prof. dr Mladen Veinović

Student:

Tatić Dušan

410035/2009

Beograd, 2015. godina

Aplikacija eMentor

Sažetak: U ovom projektnom zadatku objašnjeni su alati i platforme koji će se koristiti za razvoj aplikacije eMentor kao i sama specifikacija.

Ključne reči: php, mysql, cakephp, mvc, html, aplikacija.

Specification for application eMentor

Abstract: This project assignment elaborates tools and platforms that will be used to develop application eMentor, and its specification.

Keywords: php, mysql, cakephp, mvc, html, application.

Sadržaj

1	Uvod	4
1.1	Predmet istraživanja	4
1.2	Ciljevi istraživanja.....	4
1.3	Naučne metode.....	4
2	MVC	5
2.1	HTML.....	5
2.2	PHP.....	6
2.3	Realacione baze podataka	7
2.4	CakePHP	8
3	Specifikacija aplikacije eMentor	9
3.1	Nivoi korisnika	9
3.2	Funkcionalna specifikacija.....	9
3.2.1	Funkcije adminstratora	9
3.2.2	Funkcije profesora.....	10
3.2.3	Funkcije studenta	11
4	Projekat baze podataka	12
5	Razvoj aplikacije	21
5.1	Konfigurisanje okruženja	21
5.2	Model (Model)	22
5.3	Kontroler (Controller)	24
6	Zaključak	30
7	Literatura	31

1 Uvod

1.1 Predmet istraživanja

Predmet ovog istraživanja je primena MVC (Model-View-Controller) koncepta i platformi koje podržavaju ovaj koncept za razvoj eMentor aplikacije.

1.2 Ciljevi istraživanja

Naučni cilj: naučni cilj je prikaz primene savremenih tehnologija u softverskom inženjerstvu.

Društveni ciljevi: društveni cilj je automatizacija procesa komunikacije studenta i mentora u procesu izrade master rada.

1.3 Naučne metode

Pri izradi ovog rada korišćene su osnovne naučne metode deskripcije, indukcije, dedukcije, analize, sinteze i generalizacije.

Metoda deskripcije je korišćena pri objašnjavanju osnovnih pojmova i sistema funkcionisanja.

U teorijskim segmentima su primenjene metode indukcije, dedukcije i analize, a u izvođenju zaključaka metode sinteze i generalizacije.

2 MVC

MVC (Model-View-Controller) predstavlja softversku arhitekturu, osnovna ideja razvoja strukture PHP aplikacija bila je razdvajanje triju osnovnih logičkih celina aplikacija:

- 1) HTML templejta (View)
- 2) Dela PHP koda za manipulaciju podacima iz baze podataka (Model)
- 3) Dela koji vrši obradu podataka dobijenih iz baze podataka i prosledjuje ih u odgovarajuće HTML templejte (Controller)

Razdvajanje logičkog dela aplikacije (kontrolera i modela) od vizuelnog dela (view fajlova) pruža mnogobrojne pogodnosti – povećava upotrebljivost HTML templejta uz mogućnost višestrukog koriscenja istog HTML koda u okviru jedne aplikacije, omogućava dizajnerima i frontend programerima da nesmetano rade na izmeni templejta bez neophodnog poznavanja logičke strukture celokupne aplikacije i najzad, omogućava backend programerima da kreiraju funkcionalnost aplikacije bez prevelikog zadiranja u sam izgled Web stranica.

CakePHP je besplatan framework otvorenog koda (engl. open-source), za brz razvoj aplikacija (eng. rapid development framework) u PHP-u . To je skup funkcija, klasa i run-time infrastructure za programere koji kreiraju web aplikacije. Primarni cilj Cakephp-a je da omogući brz, precizan i jednostavan razvoj aplikacija baziranih na MVC arhitekturi.

2.1 HTML

HTML je skraćenica od Hyper Text Markup Language, što se može prevesti kao jezik za označavanje hipertekstova. HTML je podskup jednog šireg jezika, SGML-a (Standard General Markup Language) i koristi se za definisanje izgleda World Wide Web dokumenata, kao i za uspostavljanje veza (linkova) među dokumentima (podrazumeva se da dokument sadrži tekst, sliku, zvuk, grafiku).

HTML dokument = web strana

HTML koristi tagove da pomoću njih ukaže pretraživaču kako neki tekst ili slika treba da budu prikazani na ekranu. Tag predstavlja oznaku koja opisuje šta treba uraditi sa tekстом i slikama koji nose poruku na sajtu. Tagovi se stavljaju unutar < > zagrada. Tagovi se postavljaju na početak nekog dela dokumenta (start tag), a na kraju tog dela dokumenta postavlja se tag završetka (end tag).

Primer koda jednostavne html strane:

```
<html>
<head> <title>Prva stranica</title> </head>
  <body>
    <h1>Naslov stranice</h1>
    <p>Tekst stranice.</p>
  </body>
</html>
```

2.2 PHP

PHP (PHP: Hypertext Preprocessor) je jedan od popularnijih jezika otvorenog koda koji je pre svega fokusiran na razvoj aplikacija koje se izvršavaju na strani servera. Razvoj ovog jezika je otpočeo još 1993 godine da bi 1994 godine dobio sadašnji naziv a već 1998 je postao javno dostupan za korišćenje i besplatno preuzimanje sa svoje zvanične stranice.

PHP je svojim razvijanjem programerima nudio sve više i više opcija i mogućnosti da bi danas stigao do verzije 5.6.4 koja je najnovija stabilna verzija ovog programskog jezika. Ono što je važno za ovaj jezik jeste da on funkcioniše na svim platformama kao što to nije slučaj sa nekim ostalim programskim jezicima. Treba navesti i ogromnu popularnost koju ovaj jezik ima s obzirom da PHP za pokretanje svojih aplikacija koriste Facebook, Wikipedia kao i mnogi druge popularne aplikacije i stranice.

PHP funkcioniše na osnovu obrade podataka kojih dobija od drugih jezika. Najčešće se za to koristi HTML ali se podjednako mogu koristiti i JSON i XML. PHP se često koristi i kombinaciji sa nekom od baza podataka, najčešće MySQL.

Primer html stranice korišćenjem php jezika:

```
<html>
  <head> <title> Prva stranica </title></head>
  <body>
    <?php echo '<h1>Naslov stranice</h1>'; ?>
    <?php echo '<p> Tekst stranice.</p>'; ?>
  </body>
</html>
```

2.3 Realacione baze podataka

Relaciona baza podataka je poseban tip baze podataka kod kojeg se organizacija podataka zasniva na relacionom modelu. Podaci se u ovakvim bazama organizuju u skup relacija između kojih se definišu određene veze. Relacija se definiše kao skup n-torki sa istim atributima, definisanih nad istim domenima iz kojih mogu da uzimaju vrednosti. U relacionim bazama podataka, svaka relacija mora da ima definisan primarni ključ, koji predstavlja atribut pomoću kojeg se jedinstveno identifikuje svaka n-torka. Relacija opciono može da poseduje i spoljni ključ, preko kojeg ostvaruje vezu sa drugim relacijama.

Upravljanje ovakvim bazama podataka se realizuje preko sistema za upravljanje relacionim bazama podataka. Među najpopularnijim takvim sistemima danas su: Microsoft SQL Server, Oracle Database, MySQL i drugi. Većina tih sistema koristi upitni jezik SQL za manipulaciju podacima.

Primer relacione baze - tabele povezane preko ključa (šifra smer a) .

Шифра	Студент	Просек	ШифраСмера	Шифра смера	Назив смера
432	Марко	8.2	3	1	Смер 1
436	Јован	9.1	1	2	Смер 2
527	Бојан	7.8	2	3	Смер 3

Табела1

Табела2

Primer SQL upita

```
select Табела1.Шифра, Табела1.Студент, Табела1.Просек, Табела2.Назив смера
from Табела1 join Табела2 on Табела1.ШифраСмера = Табела2.Шифра смера where
Табела1.ШифраСмера = 2 ;
```

rezultat : 527 Бојан 7.8 Смер 3

2.4 CakePHP

CakePHP je besplatan “framework” otvorenog koda može se preuzeti na adresi:
<http://www.cakephp.org>

Organizacija fajlova u CakePHP-u

CakePHP root direktorijum sadrži tri osnovna direktorijuma:

/app

/cake

/vendors

cake - osnovne biblioteke

app - mesto gde smeštamo fajlove aplikacije koje razvijamo. Razdvojenost između app direktorijuma i cake direktorijuma daje nam mogućnost da imamo mnogo aplikacija koje dele osnovne biblioteke, a takođe nam daje priliku jednostavnog osvežavanje osnovnih biblioteka kao i aplikacija koje se razvijaju.

vendors - direktorijum u koji ubacujemo spoljašnje klase

Spisak direktorijuma

/app

/Config - Sadrži fajlove podešavanje baze podataka, “debugera”...

/Console - Kozola

/Controllers - Kontroleri

/Components - Komponente

/Models - Modeli

/Plugins - Dodaci (pluginovi)

/tmp - Koristi se za keširanje i logove

/Vendors - direktorijum spoljašnjih klasa

/Views - html fajlovi

/Elements - Elementi

/Errors - Greške

/Helpers – Pomoćni php fajlovi

/Layouts - Layout-i aplikacije

/Pages - Statičke strane

/Themend – teme za aplikaciju

/webroot - Korenski direktorijum DocumentRoot
/css - css
/files - razni fajlovi
/img - slike /js - javascript
/lib - Osnovne biblioteke
/Cake – CakePHP biblioteke

3 Specifikacija aplikacije eMentor

Aplikacija eMentor treba da omogući komunikaciju studenata i mentora u procesu kreiranja pristupnih radova i master rada, pored komunikacije potrebno je da podrži i praćenje aktivnosti po korisniku.

3.1 Nivoi korisnika

Pootrebno je predvideti tri nivoa korisnika :

- 1) **Administrator** – kriranje, izmenu i pregled korisnika sistema, profesora, studijskih programa , predmeta , pregled komunikacije (student-mentor) i radova studenata.
- 2) **Profesor** – pregled i unos revidiranih radova, komentari na radove , unos ocena.
- 3) **Student** – unos radova i komentara , zahteva za mentorstvo , verzioniranje radova , registracija

3.2 Funkcionalna specifikacija

3.2.1 Funkcije administratora

Kreiranje i izmena korisnika sistema

Omogućiti pretragu, kreiranje i menjanje korisnike sistema i to :

- a) Administrator
- b) Profesor

Kreiranje i izmena studijskih programa

Omogućiti pretragu, kriranje i izmenu studijskih programa.

Kreiranje i izmena predmeta

Omogućiti pretragu, kreiranje i izmenu predmeta kao i njihovu vezu sa pripadajućim profesorima i studijskim programima.

Pretraga studenata

Omogućiti pretragu studenata po broju indeksa, kao i pretragu studenata koji nisu pristupili aplikaciji.

Generalna pretraga aktivnosti

Omogućiti pretragu aktivnosti u sistemu u zavisnosti od tipa aktivnosti, statusa aktivnosti, tipa korisnika. Realizovati izveštaj sa rezultatima pretrage u excel formatu.

3.2.2 Funkcije profesora

Pregled zahteva za mentorstvo

Omogućiti pregled pripadajućih zahteva za mentorstvo od strane studenata.

Pregled radova

Omogućiti pregled pripadajućih radova na kojim je profesor mentor. Kod pregleda omogućiti slanje na ocenjivanje rada ukoliko rad zadovoljava standard.

Pregled radova za ocenjivanje

Omogućiti pregled pripadajućih radova koje profesor treba da oceni ili je ocenio. Pored pregleda omogućiti unos ocene.

Pregled notifikacija

Omogućiti pregled aktivnih obaveštenja za pripadajuće zahteve za obradu, ovde je potrebno realizovati pregled svih aktivnosti koje profesor treba da uradi :

- Pregled zahteva za mentorstvo
- Pregled rada
- Ocenjivanje rada

Pored aktivne liste potrebno je omogućiti i pregled realizovanih aktivnosti (istorijat).

3.2.3 Funkcije studenta

Registracija

Omogućiti registraciju i kreiranje studenata u sistemu u zavisnosti koji studijski program pohađaju.

Pregled zahteva za mentorstvo

Omogućiti pregled pripadajućih podnetih zahteva za mentorstvo od strane studenta.

Pregled radova

Omogućiti pregled i kontinualni unos pripadajućih radova u zavisnosti od statusa trenutno unušenog rada. Predvideti verzioniranje dokumenata i komentare uz svaku od revizija.

Pregled notifikacija

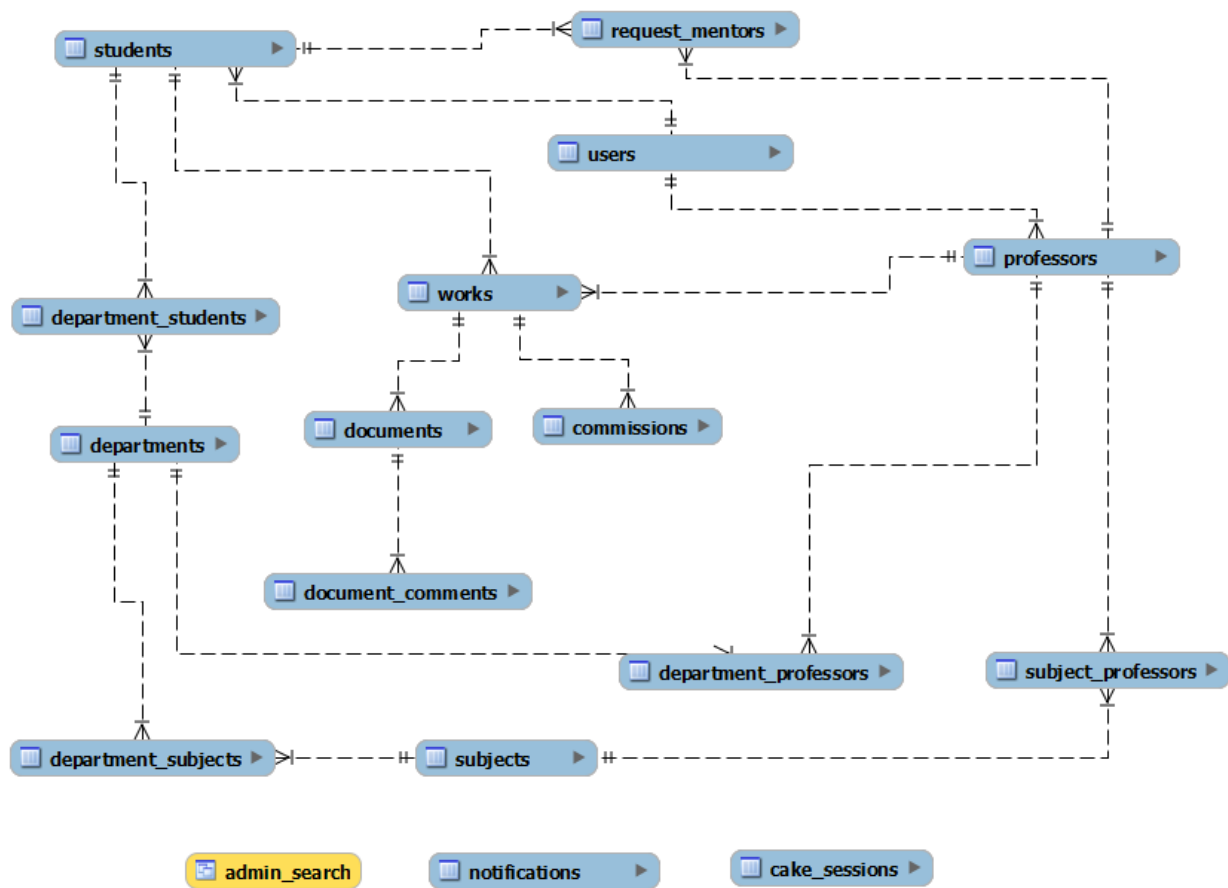
Omogućiti pregled aktivnih obaveštenja za pripadajuće zahteve za obradu , ovde je potrebno realizovati pregled svih aktivnosti koje student treba da uradi :

- Pregled odgovora na zahteva za mentorstvo
- Pregled revidiranog rada

Pored aktivne liste potrebno je omogućiti i pregled realizovanih aktivnosti (istorijat).

4 Projekat baze podataka

Za realizaciju baze podataka korišćena je MySQL. Na sledećoj slici pikazan EER dijagram modela baze.



Kod za generisanje baze podataka.

```
SET FOREIGN_KEY_CHECKS=0;

-- -----
-- Table structure for cake_sessions
-- -----
DROP TABLE IF EXISTS `cake_sessions`;
CREATE TABLE `cake_sessions` (
  `id` varchar(255) NOT NULL DEFAULT '',
  `data` text,
  `expires` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- -----
-- Table structure for users
-- -----
DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `profile_image` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `admin` tinyint(1) NOT NULL DEFAULT '0',
  `active` tinyint(1) NOT NULL DEFAULT '1',
  `tokenhash` text COLLATE utf8_unicode_ci,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  `last_login` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

-- -----
-- Records of users
-- -----
INSERT INTO `users` VALUES ('1', 'admin@mentor.com',
'$2a$10$LsEMllnKOYI.dd00xLZYOHmSqgbgMmkU6ruDzaIwauARBiY0iWVe', null, '1',
'1', null, '0000-00-00 00:00:00', '2014-12-11 09:22:28', '2014-12-11
09:22:28');
```

```

-- -----
-- Table structure for professors
-- -----
DROP TABLE IF EXISTS `professors`;
CREATE TABLE `professors` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `user_id` int(10) NOT NULL DEFAULT '0',
  `status` int(2) NOT NULL DEFAULT '0',
  `title` varchar(100) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
  `firstname` varchar(255) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT
NULL,
  `lastname` varchar(255) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT
NULL,
  `id_number` varchar(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT
NULL,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_professor_users` (`user_id`),
  CONSTRAINT `FK_professor_users` FOREIGN KEY (`user_id`) REFERENCES `users`
(`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- -----
-- Table structure for students
-- -----
DROP TABLE IF EXISTS `students`;
CREATE TABLE `students` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `user_id` int(10) NOT NULL DEFAULT '0',
  `status` int(2) NOT NULL DEFAULT '0',
  `firstname` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `lastname` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `index_number` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_student_users` (`user_id`),
  CONSTRAINT `FK_student_users` FOREIGN KEY (`user_id`) REFERENCES `users`
(`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- -----
-- Table structure for departments
-- -----
DROP TABLE IF EXISTS `departments`;
CREATE TABLE `departments` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `name` text COLLATE utf8_unicode_ci,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```

-- -----
-- Table structure for subjects
-- -----
DROP TABLE IF EXISTS `subjects`;
CREATE TABLE `subjects` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `name` text COLLATE utf8_unicode_ci NOT NULL,
  `description` text COLLATE utf8_unicode_ci NOT NULL,
  `required` tinyint(1) NOT NULL DEFAULT '0',
  `semester` int(2) NULL DEFAULT NULL,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- -----
-- Table structure for subject_professors
-- -----
DROP TABLE IF EXISTS `subject_professors`;
CREATE TABLE `subject_professors` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `subject_id` int(10) NOT NULL DEFAULT '0',
  `professor_id` int(10) NOT NULL DEFAULT '0',
  `active` tinyint(1) NOT NULL DEFAULT '1',
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_subject_professor_subjects` (`subject_id`),
  KEY `FK_subject_professor_professors` (`professor_id`),
  CONSTRAINT `FK_subject_professor_professors` FOREIGN KEY (`professor_id`)
REFERENCES `professors` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FK_subject_professor_subjects` FOREIGN KEY (`subject_id`)
REFERENCES `subjects` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- -----
-- Table structure for request_mentors
-- -----
DROP TABLE IF EXISTS `request_mentors`;
CREATE TABLE `request_mentors` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `student_id` int(10) NOT NULL DEFAULT '0',
  `professor_id` int(10) NOT NULL DEFAULT '0',
  `status` int(2) NOT NULL DEFAULT '1',
  `request` text COLLATE utf8_unicode_ci NOT NULL,
  `response` text COLLATE utf8_unicode_ci NOT NULL,
  `description` varchar(250) COLLATE utf8_unicode_ci DEFAULT NULL,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_request_mentor_student` (`student_id`),
  KEY `FK_request_mentor_professor` (`professor_id`),
  CONSTRAINT `FK_request_mentor_professor` FOREIGN KEY (`professor_id`)
REFERENCES `professors` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FK_request_mentor_student` FOREIGN KEY (`student_id`)
REFERENCES `students` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

```

```

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-----
-- Table structure for works
-----
DROP TABLE IF EXISTS `works`;
CREATE TABLE `works` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `student_id` int(10) NOT NULL DEFAULT '0',
  `professor_id` int(10) NOT NULL DEFAULT '0',
  `revision` int(3) NOT NULL DEFAULT '0',
  `type` int(2) NOT NULL DEFAULT '0',
  `status` int(2) NOT NULL DEFAULT '0',
  `name` text COLLATE utf8_unicode_ci NOT NULL,
  `description` text COLLATE utf8_unicode_ci NOT NULL,
  `mark` int(2) NOT NULL,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_work_student` (`student_id`),
  KEY `FK_work_professor` (`professor_id`),
  CONSTRAINT `FK_work_professor` FOREIGN KEY (`professor_id`) REFERENCES
`professors` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FK_work_student` FOREIGN KEY (`student_id`) REFERENCES
`students` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-----
-- Table structure for documents
-----
DROP TABLE IF EXISTS `documents`;
CREATE TABLE `documents` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `work_id` int(10) NOT NULL DEFAULT '0',
  `parent_id` int(10) NOT NULL DEFAULT '0',
  `name` varchar(250) COLLATE utf8_unicode_ci NOT NULL DEFAULT '0',
  `active` tinyint(1) NOT NULL DEFAULT '0',
  `version` int(5) NOT NULL DEFAULT '0',
  `revision` tinyint(1) NOT NULL DEFAULT '0',
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_document_work` (`work_id`),
  KEY `FK_documents_documents` (`parent_id`),
  CONSTRAINT `FK_document_work` FOREIGN KEY (`work_id`) REFERENCES `works`
(`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```

-----
-- Table structure for document_comments
-----
DROP TABLE IF EXISTS `document_comments`;
CREATE TABLE `document_comments` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `document_id` int(10) NOT NULL DEFAULT '0',
  `professor_id` int(10) NOT NULL DEFAULT '0',
  `student_id` int(10) NOT NULL DEFAULT '0',
  `description` text COLLATE utf8_unicode_ci NOT NULL,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_document_comment_document` (`document_id`),
  KEY `FK_document_comment_professor` (`professor_id`),
  KEY `FK_document_comments_students` (`student_id`),
  CONSTRAINT `FK_document_comment_document` FOREIGN KEY (`document_id`)
REFERENCES `documents` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-----
-- Table structure for commissions
-----
DROP TABLE IF EXISTS `commissions`;
CREATE TABLE `commissions` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `work_id` int(10) NOT NULL DEFAULT '0',
  `subject_id` int(10) NOT NULL DEFAULT '0',
  `professor_id` int(10) NOT NULL DEFAULT '0',
  `comment` text COLLATE utf8_unicode_ci NOT NULL,
  `mark` int(2) NOT NULL,
  `president` tinyint(1) NOT NULL DEFAULT '0',
  `status` int(2) NOT NULL DEFAULT '0',
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_commissions_professors` (`professor_id`),
  KEY `FK_commissions_works` (`work_id`),
  KEY `FK_commissions_subjects` (`subject_id`),
  CONSTRAINT `FK_commissions_works` FOREIGN KEY (`work_id`) REFERENCES
`works` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```

-----
-- Table structure for department_professors
-----
DROP TABLE IF EXISTS `department_professors`;
CREATE TABLE `department_professors` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `department_id` int(10) NOT NULL DEFAULT '0',
  `professor_id` int(10) NOT NULL DEFAULT '0',
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_department_professor_department` (`department_id`),
  KEY `FK_department_professor_professor` (`professor_id`),
  CONSTRAINT `FK_department_professor_department` FOREIGN KEY
(`department_id`) REFERENCES `departments` (`id`) ON DELETE NO ACTION ON
UPDATE NO ACTION,
  CONSTRAINT `FK_department_professor_professor` FOREIGN KEY (`professor_id`)
REFERENCES `professors` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-----
-- Table structure for department_students
-----
DROP TABLE IF EXISTS `department_students`;
CREATE TABLE `department_students` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `department_id` int(10) NOT NULL DEFAULT '0',
  `student_id` int(10) NOT NULL DEFAULT '0',
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_department_student_department` (`department_id`),
  KEY `FK_department_student_student` (`student_id`),
  CONSTRAINT `FK_department_student_department` FOREIGN KEY (`department_id`)
REFERENCES `departments` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `FK_department_student_student` FOREIGN KEY (`student_id`)
REFERENCES `students` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```

-- -----
-- Table structure for department_subjects
-- -----
DROP TABLE IF EXISTS `department_subjects`;
CREATE TABLE `department_subjects` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `subject_id` int(10) NOT NULL,
  `department_id` int(10) NOT NULL,
  `active` tinyint(1) NOT NULL DEFAULT '1',
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_department_subjects_subjects` (`subject_id`),
  KEY `FK_department_subjects_departments` (`department_id`),
  CONSTRAINT `FK_department_subjects_departments` FOREIGN KEY
(`department_id`) REFERENCES `departments` (`id`) ON DELETE NO ACTION ON
UPDATE NO ACTION,
  CONSTRAINT `FK_department_subjects_subjects` FOREIGN KEY (`subject_id`)
REFERENCES `subjects` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- -----
-- Table structure for notifications
-- -----
DROP TABLE IF EXISTS `notifications`;
CREATE TABLE `notifications` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `professor_id` int(10) NOT NULL DEFAULT '0',
  `student_id` int(10) NOT NULL DEFAULT '0',
  `request_mentor_id` int(10) NOT NULL DEFAULT '0',
  `work_id` int(10) NOT NULL DEFAULT '0',
  `document_id` int(10) NOT NULL DEFAULT '0',
  `active` tinyint(1) NOT NULL DEFAULT '0',
  `type` int(2) NOT NULL DEFAULT '0',
  `description` text COLLATE utf8_unicode_ci NOT NULL,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_notification_professor` (`professor_id`),
  KEY `FK_notification_student` (`student_id`),
  KEY `FK_notifications_works` (`work_id`),
  KEY `FK_notifications_request_mentors` (`request_mentor_id`),
  KEY `FK_notifications_documents` (`document_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```

-- -----
-- View structure for admin_search
-- -----
DROP VIEW IF EXISTS `admin_search`;
CREATE VIEW `admin_search` AS select n.id as nId, n.professor_id as n_pId,
    n.student_id as n_sId, n.request_mentor_id as n_rmId, n.work_id
as n_wId,
    n.document_id as n_dId , n.active as n_a, n.description as n_d ,
n.`type` as n_t ,
    n.created as n_c, n.modified as n_m,
    p.firstname as p_fn, p.lastname as p_ln,
    s.firstname as s_fn, s.lastname as s_ln,
    s.index_number as s_in,
    w.revision as w_r, w.`type` as w_t, w.`status` as w_s,
w.student_id w_sId , w.professor_id as w_pId
    from notifications as n left join professors as p on (n.professor_id =
p.id)
        left join students as s on (n.student_id = s.id)
        left join works as w on (n.work_id = w.id) ;

```

5 Razvoj aplikacije

Aplikacija eMenor realizovana je korišćenjem CakePHP-a struktura koda prati MVC metodologiju.

5.1 Konfigurisanje okruženja

Okruženje koje je korišćeno za razvoj aplikacije:

- 1) MySQL – 5.6.17
- 2) PHP – 5.5.12
- 3) Apache web server – 2.4.9
- 4) CakePHP - 2.5.6

Koraci za pokretanje aplikacije:

- a) Aplikaciju postavimo na u okviru „www“ direktorijuma na web serveru.
- b) U okviru MySQL servera kreiramo novu bazu

```
CREATE DATABASE `mentor` /*!40100 COLLATE 'utf8_general_ci' */
```

- c) Pustimo sql script koji se nalazi na putanji „app/Config/Schema/mentor.sql“
- d) Kofigurišemo „app/Config/database.php“

```
public $default = array(  
    'datasource' => 'Database/Mysql',  
    'persistent' => false,  
    'host' => 'localhost',  
    'login' => '###',  
    'password' => '###',  
    'database' => 'mentor',  
    'prefix' => '',  
    'encoding' => 'utf8',  
);
```

- e) Slanje email podešavamo „app/Config/email.php“

```
public $default = array(  
    'transport' => 'Smtplib',  
    'host' => '###',  
    'port' => 2525,  
    'timeout' => 30,  
    'username' => '###@###',  
    'password' => '###',  
);
```

- f) „Debug“ podesiti „app/Config/core.php“
`Configure::write('debug', 2);`

5.2 Model (Model)

Modeli služe za komunikaciju sa bazom podataka. Struktura foldera u kojoj se nalaze modeli je „app/Model/...” u okviru aplikacije eMentor imamo sledeće modele:

AdminSearch

Koristi se za pretragu notifikacija u okviru administracije, ovaj model odgovoran je za komunikaciju i prenos podataka iz sql view-a u bazi „admin_search“.

Commission

Koristi se za čuvanje podataka o ocenama i komentarima članova komisije na master radu ili po predmetima koji su objedinjeni po pristupnim radovima, stim što se samo konačna ocena master rada nalazi u „Work“ modelu.

Department

Model za čuvanje osnovnih podataka o studijskim programima.

DepartmentProfessor

Vezni model profesora i studijskog programa.

DepartmentStudent

Vezni model studenata i studijskog programa.

DepartmentSubject

Vezni model predmeta i studijskog programa.

Document

Model za čuvanje podataka o verzijama dokumenata za određeni rad.

DocumentComment

Model za čuvanje podataka o komentarima na dokumenat.

Notification

Model za čuvanje podataka o obaveštenjima o obavljenim aktivnostima profesora i studenata.

Professor

Model za čuvanje osnovnih podataka o profesorima.

RequestMentor

Model za čuvanje podataka o zahtevima za mentorstvo.

Student

Model za čuvanje osnovnih podataka o studentima.

Subject

Model za čuvanje osnovnih podataka o predmetima.

SubjectProfessor

Vezni model profesora i predmeta koje predaju.

User

Model za čuvanje osnovnih podataka o korisnicima sistema, kroz njega se vrši autentifikacija.

Work

Model za čuvanje podataka o radovima studenata.

5.3 Kontroler (Controller)

Kontroleri služe za poslovnu logiku. Struktura foldera u kojoj se nalaze kontroleri je „app/Controller/...” u okviru aplikacije eMentor imamo sledeće kontrolere:

AdminSearches

Implementirana je logika za pretragu ima 2 metode koje se mogu pozvati:

1) *admin_index*

Pretraga obaveštenja

2) *admin_excel*

Izveštaj o rezultatima pretrage u excel formatu.

Obe metode su dostupne samo administratorima sistema.

Works

Metode koje su implementirane :

1) *index*

Pregled unetih radova od strane konkretnog studenta.

2) *professor_index*

Pregled i pretraga radova za profesora na kojima je on mentor.

3) *view*

Pregled rada sa pratećim podacima prilagođen studentu.

4) *professor_view*

Pregled rada sa pratećim podacima prilagođen profesoru.

5) *add_first*

Kreiranje prvog pristupnog rada, uz odabir mentora i izbornog predmeta.

6) *add_second*

Kreiranje drugog pristupnog rada u odabir mentora i izbornog predmeta.

7) *add_master*

Kreiranje master rada uz odabir mentora.

8) *admin_view*

Pregled rada od strane administratora.

UsersController

1) *admin_index*

Pregled i pretraga kreiranih korisnika.

2) *admin_view*

Pregled korisnika od strane administratora.

3) *view*

Pregled korisničkog naloga za logovanog korisnika.

4) *login*

Prijava korisnika na sistem.

5) *admin_login*

Prijava administratora na sistem.

6) *logout*

Odjava korisnika iz sistema.

7) *admin_logout*

Odjava administratora iz sistema.

8) *admin_add*

Kreiranje korisnika od strane administratora.

9) *admin_edit*

Izmena korisnika od strane administratora.

10) *forget_password*

Zahtev za izmenu lozinke.

11) *reset_password*

Izmena lozinke.

SubjectsController

1) *admin_index*

Pregled i pretraga kreiranih predmeta, dostupan administratoru.

2) *admin_view*

Pregled pojedinačnog predmeta, dostupan administratoru.

3) *admin_add*

Unos predmeta i povezivanje sa studijskim programom, dostupan administratoru.

4) *admin_edit*

Izmena predmeta i veza sa studijskim programom, dostupan administratoru.

StudentsController

1) *admin_index*

Pregled i pretraga kreiranih studenata, dostupan administratoru.

2) *admin_no_activity*

Pregled i pretraga kreiranih studenata koji se nisu logovali na aplikaciju, dostupan administratoru.

3) *admin_view*

Pregled pojedinačnog naloga za studenta, dostupan administratoru.

4) *register*

Registracija studenata.

5) *searchStudent*

Metoda za pretragu studenata kombinovanjem parametara: Ime , Prezime , broj indeksa

Vraća listu studenata koji odgovaraju kriterijumu u *json* formatu.

RequestMentorsController

1) *index*

Pregled kreiranih zahteva za mentorstvo za prijavljenog studenta.

2) *professor_index*

Pregled kreiranih zahteva za mentorstvo za prijavljenog profesora.

3) *add*

Podnošenje zahteva za mentorstvo.

4) *professor_response*

Odgovor na podneti zahtev za mentorstvo.

5) *professor_view*

Pregled pojedinačnog zahteva za mentorstvo za profesora.

6) *view*

Pregled pojedinačnog zahteva za mentorstvo za studenta.

7) *admin_view*

Pregled pojedinačnog zahteva za mentorstvo za administratora.

ProfessorsController

1) admin_index

Pregled i pretraga kreiranih profesora, dostupan administratoru.

2) admin_view

Pregled pojedinačnog profesora, dostupan administratoru.

3) admin_add

Unos profesora i povezivanje sa studijskim programom i predmetima, dostupan administratoru.

4) admin_edit

Izmena profesora i veza sa studijskim programom i predmetom, dostupan administratoru.

5) searchProfessor

Metoda za pretragu profesora kombinovanjem parametara: Ime , Prezime
Vraća listu profesora koji odgovaraju kriterijumu u *json* formatu.

NotificationsController

1) student_index

Pregled kreiranih obaveštenja za studenta.

2) student_history_index

Pregled istorijata obaveštenja za studenta.

3) professor_index

Pregled kreiranih obaveštenja za profesora.

4) professor_history_index

Pregled istorijata obaveštenja za profesora.

DocumentsController

1) add

Metoda za kreiranje odgovarajuće verzije dokumenta.

2) sendToRevision

Metoda za slanje verzije dokumenta na reviziju.

3) downloadDoc

Metoda za „download“ odgovarajuće verzije dokumenta.

4) admin_downloadDoc

Metoda za „download“ odgovarajuće verzije dokumenta, dostupna administratoru.

DocumentCommentsController

1) rework

Metoda koja je dostupna mentoru prilikom revizije verzije rada, vraća rad na doradu studentu uz dokument revizije i komentar.

2) approve

Metoda koja je dostupna mentoru prilikom revizije prvog i drugog rada, odobrava rad i prosleđuje listi profesora koji trebaju da unesu ocene.

3) approve_master

Metoda koja je dostupna mentoru prilikom revizije master rada, odobrava rad i prosleđuje listi profesora koji su članovi komisije.

4) view_professor

Pregled komentara profesora na verziju dokumenta.

5) view

Pregled komentara studenta na verziju dokumenta.

DepartmentsController

1) admin_index

Pregled i pretraga kreiranih studijskih programa, dostupan administratoru.

2) admin_view

Pregled pojedinačnog studijskog programa, dostupan administratoru.

3) admin_add

Unos studijskog programa, dostupan administratoru.

4) admin_edit

Izmena studijskog programa, dostupan administratoru.

CommissionsController

1) index

Pregled i pretraga kreiranih zahteva za ocenjivanje (komisije) dostupan pojedinačno profesoru samo njegove zahteve.

2) view_work

Pregled pojedinačnog rada u procesu ocenjivanja, vidi se samo poslednja aktivna verzija dokumenta.

3) add_mark

Unos ocene za odgovarajući predmet za prvi i drugi pristupni rad, i ocene člana komisije za master rad.

4) add_final_mark

Unos konačne ocene na master rad dostupno predsedniku komisije.

6 Zaključak

Olakšavanjem pristupa u komunikaciji student-mentor u procesu izrade master rada, smanjuju se određene mentalne blokade i skraćuje vreme neophodno za dolazak do potrebnih informacija.

Pored olakšanog i efikasnijeg rada studenata i profesora, samo uprava obrazovne institucije ima kompletan pregled dešavanja u okviru pojedinačne komunikacije što u mnogome može da poboljša pristup u pojedinačnim slučajevima. Mogućnost izveštavanja o progresu pojedinca, kao i o zauzetosti pojedinca donose benefite rukovodstvu obrazovne institucije, kao i vlasnicima procesa, odnosno nastavnicima, jer na jednostavan i brz način mogu dobiti izveštaje o performansama svojih procesa.

Povećanjem obima obrazovnih programa, usložnjava se upravljanje ovakvom institucijom, pa bi se primenom metodologije i softvera obrađenog u ovom radu, mogli ostvariti značajni benefiti, kako na strateškom, tako i na individualnom nivou radnika i studenata, odnosno učesnika u menadžmentu institucijom i obrazovnom procesu.

7 Literatura

[1] „CakePHP“ [Na mreži]. Available: <http://cakephp.org/>

[2] „PHP.net“ [Na mreži]. Available: <http://php.net/>

[3] „HTML“ [Na mreži]. Available: <http://www.w3.org/html/>

[4] „MySQL“ [Na mreži]. Available: <http://www.mysql.com/>