

UNIVERZITET U BEOGRADU
FAKULTET ORGANIZACIONIH NAUKA

ZAVRŠNI RAD

Tema: Razvoj aplikacije za upravljanje znanjem

Mentor:

dr. Božidar Radenković

Student:

Nikola Ugrinović 63/11

Beograd, 2015

Sadržaj

1. Uvod	3
2. Tehnologije primenjene u razvoju aplikacije	4
1. Symfony 2 Framework	4
1. MVC Patern.....	5
2. FOSUserBundle	16
2. Bootstrap	20
3. Materialize	21
4. Chart JS.....	21
3. Aplikacija za upravljanje znanjem	23
1. Opis aplikacije	23
2. Korisnički zahtev	24
3. Specifikacija problema pomoću slučajeva korišćenja	24
1. Slučaj korišćenja - Registracija korisnika	26
2. Slučaj korišćenja - Prijavljivanje korisnika.....	27
3. Slučaj korišćenja - Odjavljivanje korisnika	28
4. Slučaj korišćenja - Kreiranje odeljenja	29
5. Slučaj korišćenja - Prikaz odeljenja	30
6. Slučaj korišćenja - Kreiranje zadataka	31
7. Slučaj korišćenja - Prikaz zadataka.....	32
8. Slučaj korišćenja - Izmena zadataka	33
9. Slučaj korišćenja - Brisanje zadataka.....	35
10. Slučaj korišćenja - Kreiranje članka.....	36
11. Slučaj korišćenja - Prikaz članka.....	37
12. Slučaj korišćenja - Izmena članka	38
13. Slučaj korišćenja - Pretraga članaka.....	40
14. Slučaj korišćenja - Deljenje video sadržaja	41
15. Slučaj korišćenja - Prikaz video sadržaja	42
16. Slučaj korišćenja - Deljenje linkova.....	43
17. Slučaj korišćenja - Prikaz Linkova	44
18. Slučaj korišćenja - Kreiranje kategorija	46
19. Slučaj korišćenja - Pretraga najčešćih pitanja	47

20.	Slučaj korišćenja - Pregled profila korisnika.....	48
21.	Slučaj korišćenja - Kreiranje postova sa kodom	49
22.	Slučaj korišćenja - Komentarisanje entiteta	50
4.	Analiza.....	52
1.	Dijagram klasa.....	52
2.	PMOV	53
3.	Relacioni model.....	53
5.	Korisničko uputstvo.....	54
1.	Prijavljivanje / Registracija	54
2.	Početna stranica.....	55
3.	Odeljenja	57
4.	Zadaci	58
5.	Kategorije	59
6.	Članci	60
7.	Video sadržaj.....	61
8.	Linkovi	61
9.	Pitanja.....	62
10.	Postovi sa kodom.....	63
11.	Profil	64
12.	Komentari	65
6.	Zaključak	65
7.	Literatura	66

1. Uvod

Aplikacije za upravljanje znanjem služe za prikupljanje, čuvanje kao i pružanje uvida u informacije najčešće unutar jedne organizacije. Korisne su kako u manjim tako i u većim organizacijama gde su možda i od ključnog značaja prilikom upravljanja organizacijom jer su u svakom trenutku dostupne informacije koje su neophodne. Pored olakšavanja prilikom upravljanja, aplikacije za upravljanje znanjem su jako korisne prilikom razmene informacija među članovima te organizacije, omogućavaju praćenje pristiglih informacija, kao i razmenu mišljenja članova organizacije na temu tih informacija čime se postiže neka vrsta “brainstorming” efekta. Ovaj efekat se smatra jako korisnim u organizaciji, doprinosi stvaranju većeg broja ideja i ima pozitivan uticaj na članove organizacije jer im omogućava učešće u donošenju nekih odluka.

Znanje u organizaciji može biti informacija, na primer o novim tehnologijama ili načinima rada na tržištu, video sadržaj koji prikazuje novu tehnologiju ili pokazuje način primene nekog novog alata u razvoju proizvoda, zatim članci koji sadrže korisne informacije za rad kao i najbolje prakse iz oblasti u kojoj organizacija posluje. Pored toga znanje može biti statistika, organizacija to znanje može primeniti da usmeri svoje poslovanje u pravom smeru kako bi povećala profit ili ušla na neko novo tržište.

Aplikacije za upravljanje znanjem omogućavaju razmenu članaka, video sadržaja kao i drugih raznih oblika informacija koji su od značaja za jednu organizaciju. Ova aplikacija će staviti malo više akcenat na razmenu informacija i znanja unutar organizacije koja se bavi informaciono-komunikacionim tehnologijama na internetu. Pa će shodno tome imati neke funkcionalnosti koje su karakteristične za ovaj sektor kao što je recimo razmena „Code snippet“-a, delova koda koji sačinjavaju neku funkcionalnost.

Cilj ove aplikacije je da članovima organizacije omogući stalan priliv noviteta u industriji, novih trendova ili tehnologija tako što će članovi međusobno deliti znanje koje smatraju da je novo ili da može biti korisno ostalim članovima organizacije. Biće omogućena razmena video materijala, članaka koje članovi pišu kao i njihovo komentarisanje. Takođe aplikacija će posedovati jedan segment namenjen pitanjima. Ovaj segment će koristiti Stackoverflow API i omogućiće članovima organizacije pronalaženje odgovora na najčešća pitanja.

Administrator ima i mogućnosti da grupiše članove u odeljenja kako bi oni lakše međusobno razmenjivali znanje, kao i da im dodeljuje određene zadatke čije izvršenje može da prati i time u svakom trenutku zna kakvo je stanje sa trenutnim projektima u organizaciji.

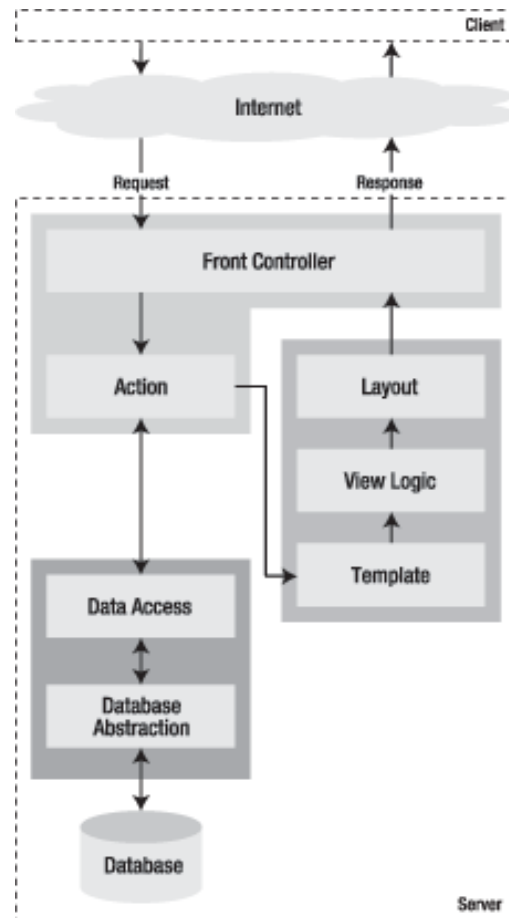
2. Tehnologije primenjene u razvoju aplikacije

1. Symfony 2 Framework

Symfony 2 Framework (u daljem tekstu Symfony) je PHP (Hypertext Preprocessor) razvojno okruženje razvijeno od strane SensioLabs kompanije. Razvijen je kako bi omogućio brži i sigurniji razvoj veb aplikacija. Sastoji se iz velikog broja gotovih PHP biblioteka koje zajedno čine moćno razvojno okruženje kako manjih tako i velikih i kompleksnih veb aplikacija.

Symfony poseduje sve karakteristične biblioteke neophodne za razvoj standardnih veb aplikacija. Ono što Symfony čini jednim od najboljih PHP okruženja je modularnost njegovih komponenti, naime moguće je koristiti samo one komponente koje su neophodne za razvoj konkretne aplikacije čime se smanjuje količina koda u jednoj aplikaciji i čini aplikaciju dosta preglednijom i jednostavnijom za održavanje. Ovo omogućavaju Symfony komponente koje se nazivaju „Vendori“. Vendori imaju ulogu da omogućе određenu funkcionalnost u aplikaciji, ono što je jako bitno naglasiti je to što su oni potpuno nezavisni međusobno i to nam omogućava gore pomenutu modularnost prilikom njihovog korišćenja.

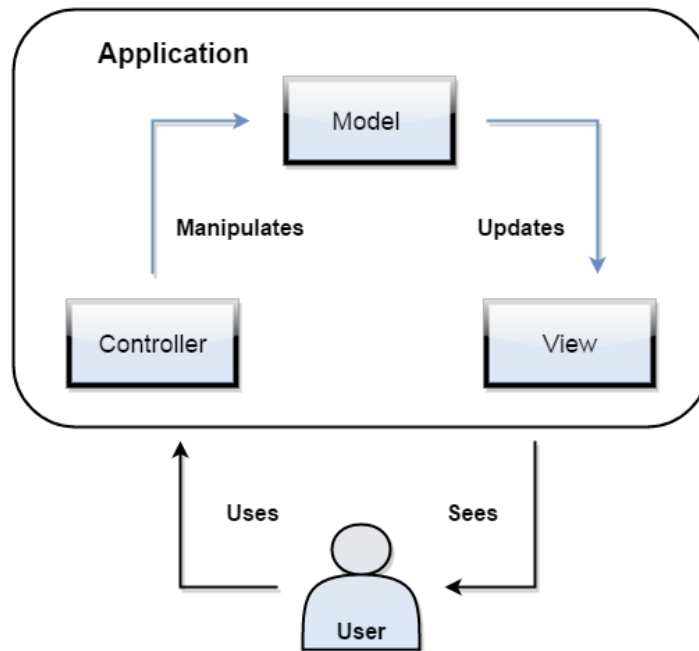
Komunikacija sa bazom se odvija preko postojećih interfejsa kao što su Propel ili Doctrine koji nam takođe omogućavaju objektno-relaciono mapiranje i kao posledicu toga omogućavaju automatsko generisanje baze podataka na osnovu modela podataka koji su kreirani u razvojnom okruženju što u velikoj meri olakšava i ubrzava rad.



Slika 1 Symfony arhitektura

1. MVC Patern

Symfony je baziran na MVC (Model – View - Contoller) paternu koji predstavlja jedan od najzastupljenijih paterna u razvoju savremenih aplikacija. Modeli podataka se najčešće mapiraju tako što naslede Symfony Entity klasu. Nakon toga kada se navedu sva polja te klase u anotacijama se navode informacije koje Doctrine koristi kako bi za nas generisao bazu podataka koja odgovara modelu koji smo kreirali. Symfony kontroleri imaju svoje akcije i svaka od njih je zadužena za odgovor na rutu koju korisnik poziva. Kontroler mora vratiti neki odgovor, odgovor može biti neka stranica ili neki od standardnih formata za razmenu podataka XML ili JSON. Symfony kao podrazumevani format za generisanje stranica koristi TWIG. TWIG je format za generisanje prikaza stranice koji omogućava vrlo lako kombinovanje klasičnog HTML koda sa PHP kodom, odnosno uklapanje promenljivih i informacija prosleđenih od strane kontrolora na stranici za prikaz, prezentacioni sloj. Ove tri Symfony komponente Entity, Controller i TWIG predstavljaju osnove MVC paterna u Symfony okruženju. Naravno izbor ovih komponenti je opcioni i okruženje može raditi sa drugim komponentama iste namene ali su ova tri rešenja najzastupljenija i iz tog razloga su korišćena u razvoju ove aplikacije.



Slika 2 Model View Controller

1. Kontroler

Kontroler predstavlja jednu od najznačajnijih komponenti aplikacije. Uloga Kontrolera je da prihvata zahtev korisnika obradi ga i vrati odgovor na taj zahtev. Kontroleri se u Symfony okruženju nazivaju imenima koja imaju sufiks “Controller” pa ukoliko imamo kontrolera koji je zadužen za korisnike nazvaćemo ga UserController. Ovo je konvencija koja se koristi kako bi Symfony kada naiđe na tu klasu znao da je u pitanju kontroler klasa. Definisane rute na koje će kontroler odgovarati se može raditi na više načina bilo u routing.yml fajlu bilo u anotacijama unutar samih kontrolera. U ovoj aplikaciji će biti korišćene anotacije za rutiranje. Primer jednostavnog kontrolera na sledećoj slici.

```

class ArticleController extends Controller
{
    /**
     * @Route("/articles", name="article-main")
     * @return string|Symfony\Component\HttpFoundation\Response
     * @Method("GET")
     */
    public function indexAction()
    {
        $articles = $this->get('article_repository')->findAll();
        $categories = $this->get('category_repository')->findAll();
        return $this->render('Article/article-all.html.twig', array('articles'=>$articles, 'categories'=> $categories));
    }
}

```

Slika 3 Jednostavan kontroler

Iz ovog primera vidimo kako izgleda akcija kontrolera. Akcije kontrolera su zapravo ništa drugo do metode klase kontroler. Svaka od akcija je zadužena da obradi zahtev koji je stigao sa rute kojoj je dodeljena. Nazivi se dodeljuju slično kao kontrolerima ali u ovom slučaju sufiks je „Action“. Navedeni primer u anotacijama pre definisanja same akcije ima sledeće stavke `@Route` stavka koja određuje za koju rutu će akcija biti pozvana sa opcionim parametrom za ime rute kako bi se kasnije u kodu svuda ruta pozivala po imenu a ne putanji što olakšava kasniju eventualnu izmenu putanje rute jer bi se ona onda promenila samo na ovom jednom mestu umesto da se menja kroz ceo kod kuda je korišćena tako da je davanje imena rutama odlična praksa. Još jedna stavka u anotacijama koja se tiče rutiranja u ovom slučaju je `@Method` stavka, ona je opcionalna i ukoliko se izostavi akcija će biti pozivana i za POST i za GET zahteve. Nakon definisanja anotacije vrši se implementacija same akcije u ovom slučaju ona vraća iz baze podataka članke i kategorije i prosleđuje ih stranici koja prikazuje to na adekvatan način.

Ukoliko ruta prima neke parametre, recimo da želimo da prikažemo samo jedan članak, u tom slučaju se kao ulazni parametar akciji kontrolera može proslediti taj parametar i on će biti dostupan prilikom njenog izvršavanja.

```
/**
 * @Route("/article/{id}", name="article-single")
 * @param $id
 * @return string|Response
 */
public function singleArticleAction($id)
{
    $article = $this->get('article_manager')->getArticle($id);
    $comments = $this->get('comment_manager')->getComments($id);

    return $this->render('Article/article-single.html.twig', array('article'=> $article, 'comments'=>$comments));
}
```

Slika 4 Kontroler sa parametrom

U ovom primeru vidimo kako je prosleđen parametar id koji je jedinstveni identifikator članka u bazi podataka. On se koristi u metodi da bi se iz baze podataka vratio konkretni članak koji je korisnik zahtevao. Ovo je slučaj kada se korisni GET zahtev obzirom da očekujemo da je parametar koji nam je potreban prosleđen putem URL-a međutim ukoliko želimo da prosledimo podatke u telu zahteva i da koristimo POST metodu u tom slučaju se kao ulazni parametar akcije umesto promenljive navodi Request obejakt. Ovaj objekat je uvek dostupan u kontroleru ukoliko se navede kao ulazni parametar akcije. Pretpostavimo da želimo da pošaljemo POST metodom podatke ime i prezime sa vrednostima za obe ove promenljive taj zahtev bi se u akciji kontrolera obradio na način prikazan na sledećoj slici.


```

/**
 * @Route("/user/", name="user-login")
 * @param Request $request
 * @return string|\Symfony\Component\HttpFoundation\Response
 * @Method("POST")
 */
public function userNewAction(Request $request)
{
    $name = $request->request->get('name');
    $lastName = $request->request->get('lastName');

    return $this->render('Article/user-profile.html.twig', array('name'=> $name, 'lastName'=>$lastName));
}

```

Slika 5 Request kao ulazni parametar kontrolera

Na ovaj način su nam dostupni svi parametri iz zahteva korisnika bilo da je reč o POST ili GET zahtevu. Ova akcija prihvata vrednosti iz zahteva i prosleđuje ih stranici koja prikazuje profil korisnika sa njegovim podacima.

2. TWIG

Twig je ekstenzija koja omogućava ispisivanje povratne informacije od kontrolera u vidu veb stranice. Twig je podrazumevana ekstenzija u Symfony okruženju mada postoje i druge koje se mogu koristiti podjednako jednostavno. Ono što Twig čini jako dobrim načinom za prikaz informacija je to što omogućava pisanje kako HTML koda tako i CSS i Javascript koda unutar njega kao i lak pristup podacima koji su vraćeni iz kontrolera ili na neki drugi način prosleđeni stranici. Twig fajlovi se mogu, za razliku od HTML fajlova, nasleđivati i uključivati jedni u druge što doprinosi većoj preglednosti koda i nema potrebe za dupliranjem koda ukoliko se neki deo stranice ponavlja što je čest slučaj sa navigacijom ili futerom.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>{% block title %}Welcome!{% endblock %}</title>
    {% block stylesheets %}{% endblock %}
    <link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}" />
</head>
<body>
    {% include 'layout/menu.html.twig' %}
    {% block body %}
        <h2>Simple twig include!</h2>
    {% endblock %}
</body>
</html>

```

Slika 6 Uključivanje twig fajlova

Unutar twig fajlova moguće je i postavljati nove promenljive kao i vršenje nekih izmena nad pristiglim promenljivima pomoću nekih od ugrađenih filtera kojih ima jako puno. Filteri se navode nakon promenljive i označavaju se sa | nakon čega sledi naziv filtera i parametri ukoliko ih taj filter zahteva. Primer filtera koji radi nad datumima možemo pogledati na sledećoj slici. Ovde kao parametre zadajemo format u kome će datum biti ispisano tako da bi nam sledeći kod ispisao datum u formatu „May 22,2015“.

```
<span class="date right">{{ video.createdAt | date("F d, Y") }}</span>
```

Slika 7 Filter za datum

Sintaksa u twig-u je vrlo jednostavna ukoliko su u pitanju HTML, CSS ili Javascript nema nikakve razlike u odnosu na klasičan HTML dokument a kada su u pitanju promenljive prosleđene iz kontrolera sintaksa je sledeća:

- {% %} se koriste ukoliko se radi o nekoj funkciji ili logičkoj operaciji kao što su recimo if uslovi, for petlje ili nasledjivanje nekog nadređenog twig fajla
- {{ }} se koriste ukoliko se vrši ispisivanje neke promenljive ili nekih atributa nekog objekta
- {# #} se koriste za komentarisanje koda

Na sledećoj slici prikazaćemo primenu ove sintakse. Prolazićemo for petljom kroz kolekciju objekata pristiglih od kontrolera i ispisivati njihove vrednosti.

```
{% if the post is mine write post heading else write post id%}  
{% for post in posts %}  
    {% if post.author != app.getUser %}  
        <h2>This is not my post!</h2>  
        {{ post.postId }}  
    {% else %}  
        <h2>This is my post!</h2>  
        {{ post.postHeading }}  
    {% endif %}  
{% endfor %}
```

Slika 8 Twig sintaksa

Kada se radi o dodavanju slika, ili nekih drugih eksternih sadržaja kao što su video sadržaji, css stilovi ili slično twig ima posebnu funkciju koja radi sa njima. Neophodno je da se resurs nalazi u “web folderu” kako bi bio dostupan a nakon toga se vrlo lako poziva unutar twig fajla funkcijom asset().

```
<script src="{{ asset('Dashboard/js/posts.js') }}"></script>
<link rel="stylesheet" href="{{ asset('Core/css/body.css') }}" />

```

Slika 9 Twig resursi

3. Model

Model podataka sadrži kompletnu logiku aplikacije. Arhitektura modela podataka se razlikuje od slučaja do slučaja i zavisi od veličine aplikacije kao i kompleksnosti. Kao što smo ranije napomenuli Symfony se sastoji od velikog broja klasa koje su ugrađene u ovo okruženje a olakšavaju nam izradu najčešćih funkcionalnosti veb aplikacija.

Grupa klasa koji zajedno predstavljaju neku funkcionalnost u Symfony okruženju se naziva Bundle. Oni su zapravo zamišljeni tako da se mogu koristiti kao odvojeni delovi i van Symfony okruženja pa tako ukoliko vam je potrebno samo da imate Request i Response objekte u svojoj aplikaciji a nije vam potrebno ništa više od Symfony okruženja možete vrlo lako samo njih izolovati i koristiti bez potrebe za celim okruženjem. Jedan bundle koji se jako često koristi je DoctrineORM. Ovaj bundle omogućava objektno relaciono mapiranje tačnije uspostavlja mapiranje između klasa i tabela u bazi podataka. Tako da svaki entitet koji poseduje aplikacija ima odgovarajuću tabelu u bazi podataka sa svim njenim poljima i vezama sa drugim entitetima. Na osnovu toga Symfony okruženje je sposobno da samo kreira bazu podataka koja odgovara modelu i da prilikom svake izmene modela primeni te izmene nad bazom podataka.

Entity

Svi entiteti u aplikaciji koje želimo da mapiramo u bazu podataka treba da se nalaze u folderu Entity. Nakon što smo naveli naziv klase i sve njene attribute sledi navođenje anotacija koje nam obezbeđuju mapiranje entiteta u bazu podataka. Anotacije sadrže tip promenljive kao i neke dodatne informacije kao što je autoincrement ili nullable. Primer jednog entiteta vidimo na sledećoj slici.

```

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass="AppBundle\Entity\Repository\ArticleRepository")
 */
class Article{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $articleId;

    /**
     * @ORM\Column(type="string")
     */
    protected $heading;

    /**
     * @ORM\Column(type="string", length=999999)
     */
    protected $content;

    /**
     * @ORM\Column(type="string", nullable=true)
     */
    protected $tags;

    /**
     * @ORM\Column(type="boolean")
     */
    protected $private;
}

```

Slika 10 Mapiranje entiteta

Na ovom primjeru vidimo kako se mapira identifikator u bazi podataka pomoću anotacija i ostalih polja u tabeli zajedno sa nekim ograničenjima pa tako kod string promenljivih možemo zadati dužinu ili nullable atribut. Kao što vidimo pre imena klase u anotacijama naveli smo naziv Repository klase (više reči o ovoj klasi biće u daljem tekstu) koju ovaj entitet koristi ukoliko se to ne navede koristiće se podrazumevana Repository klasa koju Symfony obezbeđuje za sve entitete. Pored prostih atributa koje mapiramo na ovaj način možemo mapirati i veze sa drugim entitetima.

Veze sa drugim entitetima se mapiraju na malo drugačiji način korišćenjem nekih od predefinisanih Doctrine tipova veza (OneToMany,ManyToOne, ManyToMany). Iako se u bazi podataka čuva samo jedinstveni identifikator polja u entitetima čuvaju cele objekte tako da u svakom momentu možemo pristupiti svim poljima tog entiteta kao i entitetima koji su sa njim u vezi.

```

/**
 * @ORM\ManyToOne(targetEntity="AppBundle\Entity\User", inversedBy="articles")
 * @ORM\JoinColumn(name="user_id", referencedColumnName="id")
 */
protected $userId;

/**
 * @ORM\ManyToOne(targetEntity="AppBundle\Entity\Category", inversedBy="articles")
 * @ORM\JoinColumn(name="category_id", referencedColumnName="categoryId")
 */
protected $categoryId;

/**
 * @ORM\OneToMany(targetEntity="AppBundle\Entity\Comment", mappedBy="articleId")
 */
protected $comments;

/**
 * @ORM\OneToMany(targetEntity="AppBundle\Entity\Notification", mappedBy="article")
 */
protected $notifications;

/**
 * @ORM\Column(type="string")
 */
protected $postType;

```

Slika 11 Mapiranje veza entiteta

Entity Manager

Entity Manager je sloj koji se nalazi između entiteta i Entity repository klase koja je zapravo broker baze podataka. Zadatak Entity Manager-a je da obavlja skoro svu logiku nad podacima. Na taj način se ove operacije izostavljaju iz koda kontrolera koji bi trebalo da ima samo ulogu preusmeravanja i prosleđivanja informacija na prezentacioni sloj. Kreiranje sopstvenih menadžer klasa je jedna od odličnih praksi koja se predlaže u „Symfony Best Practices“ koji predstavlja predviđene i najbolje načine za rad u ovom okruženju. Kreiranjem menadžer klase logika se izmešta iz kontrolera i obavlja u ovim klasama, takođe pribavljanje svih entiteta koji su neophodni u toj konkretnoj logičkoj operaciji se radi direktno iz ove klase tako što se koristi koncept „inject“-ovanja. Ovo je jedan od osnovnih koncepata koji se koriste u ovom okruženju kao i u većini drugih razvojnih okvira („frameworka“). Ovaj koncept se odvija tako što se klasa koji želimo da koristimo prosleđuje kao ulazni parametar konstruktora i kao argument prilikom definisanja menadžer klase kao servisa. Ovime postićemo lako pribavljanje svih entiteta neophodnih za izvršenje logičkih operacija. Na sledećoj slici videćemo primer ovog koncepta gde se unutar menadžera članaka koristi menadžer kategorije.

```

<?php
namespace AppBundle\Entity\Manager;
use ...

class ArticleManager {

    protected $repository;
    protected $categoryManager;

    public function __construct(ArticleRepository $repository, CategoryManager $categoryManager) {
        $this->repository= $repository;
        $this->categoryManager= $categoryManager;
    }
}

```

Slika 12 Menadžer članaka

```

# services.yml
article_manager:
    class: AppBundle\Entity\Manager\ArticleManager
    arguments: ['@article_repository', '@category_manager']

```

Slika 13 Argumenti prosleđeni servisu

Na ove dve slike vidimo kako se prosleđeni argumenti u servisu prihvataju kao ulazni parametri konstruktora i mogu se koristiti unutar menadžer klase.

Kada su menadžeri na ovaj način prosleđeni pribavljanje entiteta se odvija jako jednostavno pozivanjem metoda menadžer klase koje pozivaju Repository klasu koja će iz baze podataka vratiti tražene podatke.

```

public function save(Article $article, $categoryId) {
    $category= $this->categoryManager->getCategoryById($categoryId);
    $category->addArticle($article);
    $article->setCreatedAt(new \DateTime(date('Y-m-d H:i:s')));
    $this->repository->save($article);
}

```

Slika 14 Cuvanje članaka

Kao što se vidi na primeru u jednoj liniji smo vratili kategoriju iz baze podataka. Nakon toga smo dodelili članak toj kategoriji i pozvali metodu repository klase koja je odgovorna za operacije nad bazom podataka. Nakon ovog objašnjenja vidimo da je pravljenje sopstvenih Entity Manager klasa jako dobra praksa jer imamo potpunu kontrolu nad kodom kao i dosta pregledniji kod. U slučaju da smo koristili predefinisane Entity Manager i Entity Repository klase sav ovaj kod oko prevezivanja entiteta i pronalaženja iz baze bi morao da se nađe u kontroleru što bi jako zakomplikovalo kod i kontroleri bi pored svoje osnovne funkcije da preusmeravaju zahtev i vraćaju odgovor na prezentacioni sloj imali ulogu da vrše poslovnu logiku aplikacije kao i da vraćaju entitete iz baze podataka a to bi bila loša praksa.

Entity Repository

Repository klase su klase koje su namenjene komunikaciji sa bazom. Svaki entitet ima predefinisano Entity Repository klasu koju može koristiti bez potrebe da se kreira nova. Unutar te klase nalaze se predefinisane metode koje rade čuvanje entiteta u bazu, vraćanje svih entiteta, ili jednog entiteta na osnovu primarnog ključa ili bilo kog drugog atributa. To su metode `findAll()`, `findOneBy()`, `save()` i tako dalje. Međutim pravljenje sopstvenih repository klasa koje nasleđuju ovu opštu predefinisano klasu je jako dobra praksa i preporučuje se pogotovo prilikom kreiranja kompleksnijih aplikacija. Kao i menadžer klase i ove klase se najčešće implementiraju kao servisi radi lakšeg korišćenja. Kada se nasledi opšta repository klasa mogu se, pored onih predefinisanih metoda, koristiti i metode koje korisnik sam kreira u zavisnosti od potreba aplikacije. Ono što ovu klasu čini jako dobrom i fleksibilnom je što ukoliko nema potrebe za nekim složenijim upitima korisnik vrlo lako može izvršavati osnovne operacije nad bazom koje možemo videti na sledećoj slici.

```
/**
 * @param $id
 * @return mixed
 */
public function getArticle($id){
    return $this->findByArticleId($id);
}
```

Slika 15 Vraćanje članka iz baze po primarnom ključu

Ovde vidimo koliko se jednostavno vraća članak iz baze na osnovu primarnog ključa koji se prosleđuje kao ulazni parametar metode. Naravno ovo je jedna od prostih operacija nad bazom ali ukoliko nam je potrebno da izvršimo neku složeniju operaciju Symfony nam daje potpunu slobodu u tome pa tako možemo pisati upite direktno u SQL-u (Structured Query Language) ali isto tako u DQL-u (Doctrine Query Language).

Pored opšte poznatog jezika za upite nad bazom podataka SQL, Doctrine nam omogućava pisanje upita pomoću DQL-a koji umesto sa redovima u tabelama radi sa objektima. Svaki upit koji se piše piše se na osnovu atributa objekata koji se koriste u upitu. Na taj način je mnogo lakše pisanje upita i ne moramo da vodimo računa o nazivima kolona u tabelama u bazi. Naravno DQL ima podršku za sve one funkcije koje imamo u SQL-u kao što su `where`, `orderBy`, `groupBy` i tako dalje. Kreiranje upita DQL-om se postiže tako što se iz menadžer klase poziva metoda `createQueryBuilder`, ova metoda vraća objekat klase `QueryBuilder` koji se koristi za kreiranje upita u DQL-u. Nakon što smo dobili referencu na ovaj objekat možemo početi sa pozivanjem njegovih metoda i na taj način graditi naš upit. Ono što je jako dobro je to što se ove metode ne moraju pozivati u nekom predefinisanom redosledu kao što bi pisali upit u SQL-u gde na primer prvo pišemo `select` ključnu reč pa nakon nje `from` pa tek na kraju `where` ako imamo uslove tog

upita. Kod ovog načina kreiranja upita nije bitan redosled pisanja ovih metoda već se upit konstruiše na osnovu svih njih tek u vreme izvršavanja.

Pretpostavimo da je `$qb` promenljiva koja sadrži referencu na objekat klase `QueryBuilder`. Na sledećoj slici možemo videti kako se konstruiše upit korišćenjem objekta ove klase. U ovom primeru vraćamo sve članke određenog korisnika sa još jednim parametrom koji ograničava broj vraćenih članaka.

```
/**
 * @param $id
 * @param $limit
 * @return array
 */
public function findArticlesByUserLimited($id, $limit){
    $em = $this->getEntityManager();
    $qb = $em->createQueryBuilder();
    $qb->select('a')
        ->from('Article', 'a')
        ->where('a.userId = ?1')
        ->orderBy('a.createdAt', 'DESC')
        ->setParameter('1', $id)
        ->setMaxResults($limit);
    return $qb->getQuery()->getResult();
}
```

Slika 16 Primer DQL-a

Kao što vidimo na primeru nakon kreiranja objekta pozivamo metode koje umesto nas generišu upit koji će se izvršiti. Uočavamo standardne SQL ključne reči `select`, `where`, `orderBy` ali i `setParameter` metodu. Ova metoda poboljšava zaštitu tako što sprečava „SQL injection“ napade time što parametre upita upisuje u vreme izvršavanja upita čime onemogućava bilo kakvu nepredviđenu manipulaciju upitima. Takođe primećujemo i metodu `setMaxResults` koja je analogna SQL-ovom `LIMIT`.

Ono što nismo napomenuli je da ukoliko pozovemo neku od generičkih metoda `findAll`, `findBy` i slično ono što dobijamo nije samo taj entitet iz baze koji smo tražili već uz njega dobijamo i reference na objekte koji su sa njim u vezi umesto samo primarnog ključa tog entiteta. Iako nismo eksplicitno naglasili JOIN tabela Symfony u pozadini radi taj deo posla za nas. Na prvi pogled ovo je odlična funkcionalnost ali ukoliko malo razmislimo da je korisnik u većini aplikacija povezan sa dosta entiteta možemo doći u situaciju da nam jedan upit kao rezultat vrati veliki deo baze podataka kao rezultat. Ako posmatramo sa tog aspekta ovo može biti veliki problem prilikom optimizacije aplikacije.

Prost primer vraćamo korisnika iz baze podataka a korisnik je autor nekih članaka mi vraćamo i te članke koji su vezani za tog korisnika, članci imaju komentare koji su takođe vezani za neke druge korisnike i tako dalje dolazimo do činjenice da ovo može predstavljati veliki problem. Međutim ovo je vrlo dobro rešeno konceptom „lazy loading“. Ono što ovaj koncept radi je da postavlja granicu do koje se taj generički upit prostire u smisli koliko nivoa veza će se vratiti iz baze podataka. Podrazumevana vrednost tih nivoa je 3 međutim može se menjati po potrebi. Ono što je još jako bitno je da ni jedan objekat vraćen na ovaj način, osim onog koji smo zapravo tražili u ovom slučaju korisnik, neće biti inicijalizovani sve dok nam ne zatreba odnosno dok ne

pokušamo da pristupimo nekom od njegovih atributa čime se upravo rešava onaj problem kod optimizacije. U malim aplikacijama ovo ne bi trebalo da predstavlja problem međutim u nekim većim i kompleksnijim može biti veliki problem.

Preporuka je da bi se takvi problemi izbegli da se prilikom pisanja upita pišu sopstveni upiti ali tako da se navedu samo oni atributi objekata, odnosno one kolone u tabeli u bazi podataka, koji su nam neophodni za dalje manipulisanje. Pa tako iako se radi join nad 3 ili više tabela upit će se jako brzo izvršiti i naša aplikacija neće imati problema sa optimizacijom.

```
/**
 * @param User $user
 * @return array
 */
public function getNotifications(User $user){
    $qb = $this->createQueryBuilder('n');
    $qb->select('n.id, a.articleId, a.heading as article_heading, a.createdAt, u.username, img.path, v.heading as video_heading, v.videoId');
    $qb->leftJoin('n.article', 'a')
        ->leftJoin('n.user', 'u')
        ->leftJoin('u.profileImg', 'img')
        ->leftJoin('n.video', 'v')
        ->where('a.userId = '.$user->getId())
        ->orWhere('v.userId = '.$user->getId())
        ->andWhere('n.seen = 0');
    return $qb->getQuery()->getResult();
}
```

Slika 17 Primer pravilnog select upita

2. FOSUserBundle

FISUserBundle je jedan od najzastupljenijih bundle-ova koji se koriste u Symfony okruženju. On rešava jednu bitnu funkcionalnost a to je upravljanje korisnicima i privilegijama u aplikaciji. Ovaj bundle je razvijen od strane Friends of Symfony zajednice koja je veoma cenjena u krugu korisnika ovog okruženja. Pored ovog projekta oni razvijaju i mnoge druge projekte koji se mogu koristiti u Symfony okruženju. Ovaj bundle rešava problem logovanja, registrovanja korisnika ali i pruža mogućnost upravljanja rolama korisnika. Kada uspostavimo ovaj bundle dobijamo gotovu klasu korisnika sa svim neophodnim atributima kao što su korisničko ime, lozinka i ostale standardne informacije o korisniku.

```

* @author Johannes M. Schmitt <schmittjoh@gmail.com>
*/
abstract class User implements UserInterface, GroupableInterface
{

    /**
     * @var string
     */
    protected $username;

    /**
     * @var string
     */
    protected $usernameCanonical;

    /**
     * @var string
     */
    protected $email;

    /**
     * @var string
     */
    protected $emailCanonical;

    /**
     * @var boolean
     */
    protected $enabled;

    /**
     * The salt to use for hashing
     *
     * @var string
     */
    protected $salt;

    /**
     * Encrypted password. Must be persisted.
     *
     * @var string
     */
    protected $password;

    /**
     * Plain password. Used for model validation. Must not be persisted.
     *
     * @var string
     */
}

```

Slika 18 Osnovna korisnik klasa

Naravno uglavnom naša aplikacija zahteva neku dodatnu informaciju o korisniku ali se to vrlo lako rešava nasleđivanjem ove dobijene klase svojom i dodavanjem potrebnih polja.

```

/**
 * @ORM\Entity(repositoryClass="AppBundle\Entity\Repository\UserRepository")
 * @ORM\Table(name="User")
 */
class User extends BaseUser{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @ORM\Column(type="string")
     */
    protected $name;

    /**
     * @ORM\ManyToOne(targetEntity="AppBundle\Entity\Department", inversedBy="employees")
     * @ORM\JoinColumn(name="department_id", referencedColumnName="departmentId")
     */
    protected $departmentId;

    /**
     * @ORM\OneToMany(targetEntity="AppBundle\Entity\Article", mappedBy="userId")
     */
    protected $articles;

    /**
     * @ORM\OneToMany(targetEntity="AppBundle\Entity\Video", mappedBy="userId")
     */
    protected $videos;

    /**
     * @ORM\OneToMany(targetEntity="AppBundle\Entity\CodeSnippet", mappedBy="userId")
     */
    protected $codeSnippets;
}

```

Slika 19 Klasa korisnik sa novim atributima

Osim ove klase ono što je još potrebno uraditi je podesiti security.yml fajl kako bi on koristio ovaj bundle. Takođe u ovom fajlu se definišu delovi aplikacije kojima korisnik može pristupiti na osnovu uloge koju poseduje. Primer ovog fajla videćemo na sledećoj slici.

```

# http://symfony.com/doc/current/book/security.html
security:
    # http://symfony.com/doc/current/book/security.html#encoding-the-user-s-password
    encoders:
        FOS\UserBundle\Model\UserInterface: sha512

    # http://symfony.com/doc/current/book/security.html#hierarchical-roles
    role_hierarchy:
        ROLE_ADMIN:       ROLE_USER
        ROLE_SUPER_ADMIN: [ROLE_USER, ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]

    # http://symfony.com/doc/current/book/security.html#where-do-users-come-from-user-providers
    providers:
        fos_userbundle:
            id: fos_user.user_manager

    # the main part of the security, where you can set up firewalls
    # for specific sections of your app
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
            # disables authentication for assets and the profiler, adapt it according to your needs
        main:
            pattern: ^/
            form_login:
                check_path: fos_user_security_check
                login_path: fos_user_security_login
                default_target_path: homepage
                provider: fos_userbundle
                use_referer: true
            logout:          true
            anonymous:        true
            # secures part of the application

    # with these settings you can restrict or allow access for different parts
    # of your application based on roles, ip, host or methods
    # http://symfony.com/doc/current/cookbook/security/access_control.html
    access_control:
        - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/register, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/resetting, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/admin/, roles: ROLE_ADMIN }
        - { path: ^/, roles: ROLE_USER }

```

Slika 20 Security.yml fajl

Vidimo u delu „access_control“ koje su privilegije odnosno delovi aplikacije kojima određeni korisnici mogu pristupiti. Stranicama login i register mogu pristupiti svi korisnici aplikacije, admin delu aplikacije može pristupiti samo korisnik sa ulogom administratora a svim ostalim delovima aplikacije korisnici koji su ulogovani na sistem bez obzira na ulogu.

2. Bootstrap

Bootstrap je CSS razvojni okvir namenjen bržem i lakšem kreiranju izgleda stranica. Razvijen od strane Twitter-a usled povećanja broja korisnika koji su pretraživali internet preko svojih mobilnih telefona. Naime veliki broj sajtova nije bio prilagođen mobilnim ekranima tako da je to osnovni cilj nastanka Bootstrap-a jer su ga i oni okarakterisali kao “mobile first”, odnosno prvenstveno namenjen mobilnim uređajima. Ono što nam on omogućava je takozvani “responsive” dizajn. To podrazumeva veliku prilagođenost sajta kako velikim tako i malim ekranima na mobilnim uređajima.

Ono što omogućava responzivnost je način na koji Bootstrap sagledava stranicu kao sadržaj raspoređen u 12 kolona. Ovaj broj je odabran iz razloga što je to broj celobrojno deljiv sa 2,3,4,6 i kao takav je veoma pogodan za deljenje sadržaja na ekranima po kolonama. Bootstrap funkcioniše tako što se HTML elementima dodeljuju klase koju se već stilizovane i na taj način štede vreme koje bi se utrošilo na stilizovanje izgleda stranice. Neke od osnovnih klasa su container, container-fluid, col, btn i ostale. Klasa container je klasa koja centrira sadržaj koji se nalazi unutar nje. Naravno svaki container se može kasnije podeliti na više kolona pomoću klase col. Ova klasa je specifična jer se javlja u nekoliko oblika u kombinaciji sa xs, sm, md i lg.

```
/* Extra small devices (phones, less than 768px) */
/* No media query since this is the default in Bootstrap */

/* Small devices (tablets, 768px and up) */
@media (min-width: @screen-sm-min) { ... }

/* Medium devices (desktops, 992px and up) */
@media (min-width: @screen-md-min) { ... }

/* Large devices (large desktops, 1200px and up) */
@media (min-width: @screen-lg-min) { ... }
```

Copy

Slika 21 Bootstrap veličine ekrana

Gde svaka od ovih oznaka označava veličinu ekrana od najmanjeg xs do najvećeg lg. Nakon ovog dela koji označava veličinu ekrana sledi broj od 1 do 12 koji označava koliko kolona, od onih 12 koje smo pomenuli, će naznačeni element zauzeti. Tako napisane HTML elemente grupišemo u elemente sa klasom row pri čemu zbir kolona unutar elemenata klase row bi trebalo da bude uvek 12. Neke od primera kombinovanja ovih kolona možemo videti na sledećoj slici preuzetoj sa oficijalnog Bootstrap sajta.

```

<div class="row">
  <div class="col-md-8">.col-md-8</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-6">.col-md-6</div>
  <div class="col-md-6">.col-md-6</div>
</div>

```

Slika 22 Primer Bootstrap kolona

3. Materialize

Materialize je Css i Javascript biblioteka koja je zasnovana na Bootstrap-u a potpuno se uklapa i omogućava korišćenje najnovijeg trenda promovisanog od strane Google-a a to je “Material design”. Ovaj način prostog i jednostavnog ali atraktivnog dizajna promovisao je Google a vrhunac popularnosti dostiže izlaskom Android 5.0 Lollipop operativnog sistema koji je kompletno baziran na ovom principu dizajna.

Ova biblioteka pored Css stilova koji su rađeni po uzoru na Bootstrap sadrži i veliki broj pomoćnih funkcionalnosti. Neke od njih su Toast poruke po uzoru na one koje se koriste na Android operativnom sistemu gde se korisniku prikazuje neka informacija, recimo da je uspešno uneo neki entitet i nakon nekoliko sekundi ona nestaje. Takođe tu su i popularni Bootstrap elementi poput Accordion elementa koji grupiše nekoliko manjih elemenata i prikazuje opširnije informacije o jednom od njih koji korisnik izabere dok sakriva za ostale elemente. Takođe koncept kartica koji je takođe zastupljen u Google-ovim servisima gde se po karticama grupišu manje logičke celine u aplikaciji radi veće preglednosti.

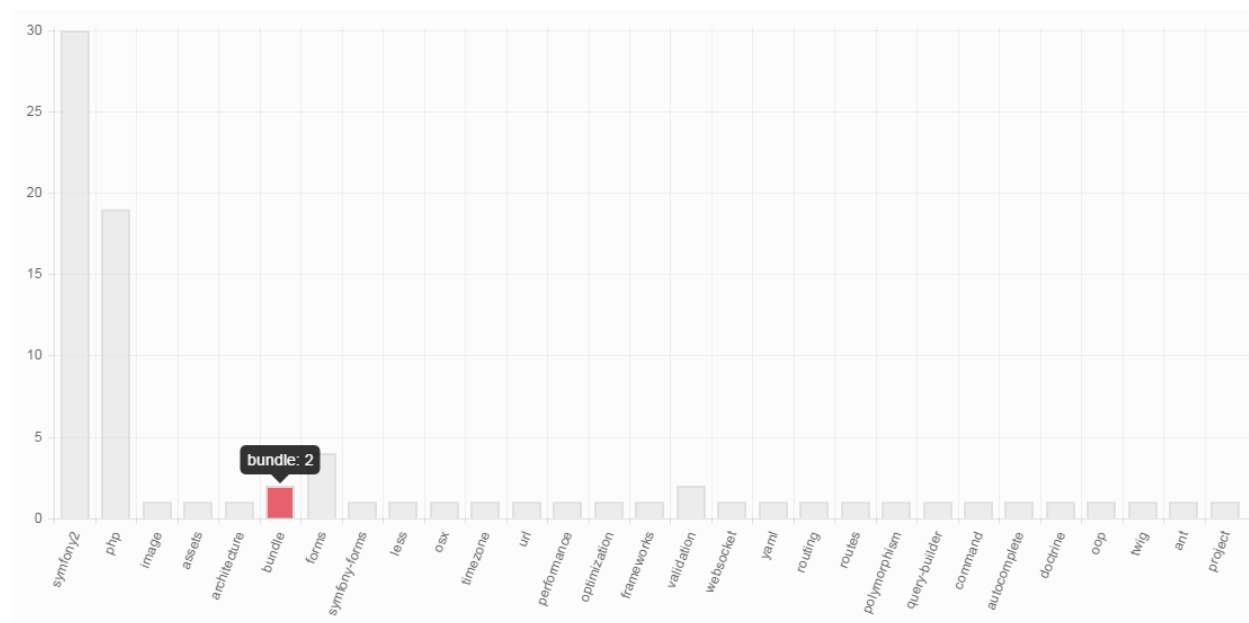
4. Chart JS

Chart JS je javascript biblioteka koja se koristi za vizuelizaciju podataka. Vrlo je jednostavna za korišćenje i na jako dobar način prikazuje podatke. Postoji više različitih tipova grafika koje možemo koristiti iz ove biblioteke kao što su Radar Chart, Polar Area Chart, Bar Chart, Line Chart i drugi.

Sve što treba da uradimo da bi smo koristili ovu biblioteku je da uključimo u projekat javascript fajlove koji su neophodni za njen rad i da prosledimo informacije nad kojima će se konstruisati grafici. Primeri grafika korišćenih u aplikaciji su na sledećim slikama.



Slika 23 Radar grafik



Slika 24 Bar chart grafik

3. Aplikacija za upravljanje znanjem

1. Opis aplikacije

KnowledgeMS je aplikacija namenjena razmeni i deljenju znanja unutar organizacije. Ova aplikacija je više bazirana na primenu u oblasti informaciono komunikacionih tehnologija na internetu. Aplikacija omogućava logovanje korisnika na sistem i nakon toga mogućnost kreiranja ili deljenja određenog sadržaja. Administrator ima privilegije da kreira odeljenja unutar organizacije i da raspoređuje korisnike unutar tih odeljenja. Nakon toga kada se neki sadržaj kreira korisnik bira da li će on biti vidljiv na nivou cele organizacije ili samo na nivou njegovog odeljenja. Takođe administrator ima pravo kreiranja i dodeljivanja zadataka ostalim članovima organizacije. Zadatak može biti dodeljen jednom ili više članova, sadrži opis zadatka kao i vremenski period u kome bi trebalo da se izvrši određeni zadatak.

Pored ovih funkcionalnosti korisnici aplikacije imaju mogućnost kreiranja članaka koji se razvrstavaju po kategorijama. Prilikom kreiranja članaka korisnik unosi, pored kategorije, i tagove za koje je taj članak vezan čime se dobija na semantici. Takođe ukoliko korisnik želi porediti članak u okviru koga se nalazi programski kod to vrlo lako može učiniti pomoću dodatnih opcija sa strane. Taj deo teksta koji je na taj način označen kao kod biće adekvatno prikazan i formatiran kao da se radi o nekom tekstualnom editoru. Ovo omogućava pregledniji prikaz programskog koda koji bi inače bio jako teško razumljiv da nije formatiran. Korisnici takođe mogu komentarisati članke drugih korisnika. Ukoliko je neki korisnik komentarisao neki članak autoru tog članka stiže obaveštenje o tome.

Pored dodavanja članaka korisnici imaju mogućnost deljenja video sadržaja sa Youtube sajta. Deljenje video sadržaja je jako slično deljenju članaka takođe je neophodno uneti tagove kao i naslov video sadržaja. Video sadržaj se prikazuje unutar iframe elementa i koristi podrazumevani Youtube video player. Video sadržaj se takođe može komentarisati na isti način kao i članci.

Korisnik ima pristup svom profilu gde može pogledati neke osnovne informacije kao što su koliko je članaka dodao, koliko je napisao komentara i slično. U ovom delu korisnik može postaviti svoju profilnu fotografiju. Na ovoj stranici korisnik takođe ima izlistane zadatke koji su mu dodeljeni tako da ima neku vrstu podsetnika.

Sledeća funkcionalnost se odnosi na pisanje koda direktno u aplikaciji. Kako smo u uvodu rekli da je ova aplikacija malo više namenjena IT sektoru ovo bi bila jedna od funkcionalnosti koja to potvrđuje. Naime u ovom delu aplikacije korisnici mogu unutar aplikacije kucati HTML, CSS i Javascript kod i gledati uživo prikaz tog koda koji su otkucali bez potrebe da se kreiraju odvojeni dokumenti u nekim tekst editorima i da se koristi internet pregledač koji bi morao da se osvežava nakon svake promene u kodu. Ova stranica se sastoji iz četiri dela, u prvom delu se nalazi HTML, u drugom CSS editor, u trećem Javascript editor a u poslednjem prikaz trenutno otkucanog koda. Svaki od ovih editora je potpuno funkcionalan u smislu da vrši formatiranje kao i asistenciju prilikom kucanja koda ("Autocomplete"). Sve promene se izvršavaju trenutno kada je u pitanju HTML i CSS ali ako se koristi Javascript kod u tom slučaju se klikom na dugme

pokreće skripta. Deo koji služi za prikaz koda ima i mogućnost prikazivanja koda na manjim ekranima pa tako imamo predefinisane opcije za mobilne telefone, tablete, računare kao i za veće ekrane.

Takođe aplikacija ima mogućnost pretraživanja odgovora na najčešća pitanja. Ova pretraga koristi bazu podataka popularnog sajta <http://stackoverflow.com/>. Naime korisnik unosi u polje predviđeno za pitanje ono što ga zanima i nakon pretrage dobija ponuđene konkretne probleme na tu temu zajedno sa putanjom ka tačnom odgovoru. Takođe korisnik dobija statistiku o tome koji su sve tagovi najčešće korišćeni na tim temama, koliko ima odgovora na svaki od problema kao i još neke podatke vezane za pitanja i odgovore.

2. Korisnički zahtev

Potrebno je projektovati i implementirati aplikaciju koja će omogućiti upravljanje znanjem unutar organizacije. Glavne funkcionalnosti koje je potrebno obezbediti su:

- Kreiranje i razmena članaka
- Razmena video materijala
- Razmena linkova
- Razmena delova koda ("Code snippet")
- Komentisanje deljenog materijala
- Pretraga najčešćih pitanja
- Grupisanje korisnika u odeljenja

Zahtev je kreiranje aplikacije sa navedenim funkcionalnostima koja bi olakšala razmenu informacija među korisnicima unutar organizacije. Takođe omogućiće pregled profila korisnika gde će biti prikazani svi članci, video zapisi i linkovi koje je taj korisnik podelio.

3. Specifikacija problema pomoću slučajeva korišćenja

Na osnovu korisničkog zahteva uočeni su sledeći slučajevi korišćenja:

1. Registracija korisnika
2. Prijavljivanje korisnika
3. Odjavljivanje korisnika
4. Kreiranje odeljenja
5. Prikaz odeljenja
6. Kreiranje zadataka
7. Prikaz zadataka
8. Izmena zadataka
9. Brisanje zadataka
10. Kreiranje članaka
11. Prikaz članaka

12. Izmena članaka
13. Pretraga članaka
14. Deljenje video sadržaja
15. Prikaz video sadržaja
16. Deljenje linkova
17. Prikaz linkova
18. Kreiranje kategorija
19. Pretraga najčešćih pitanja
20. Pregled profila korisnika
21. Kreiranje postova sa kodom ("Code snippet")
22. Komentarisanje entiteta



Slika 25 Uml dijagram slučajeva korišćenja

1. Slučaj korišćenja - Registracija korisnika

Naziv SK

Registracija korisnika

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

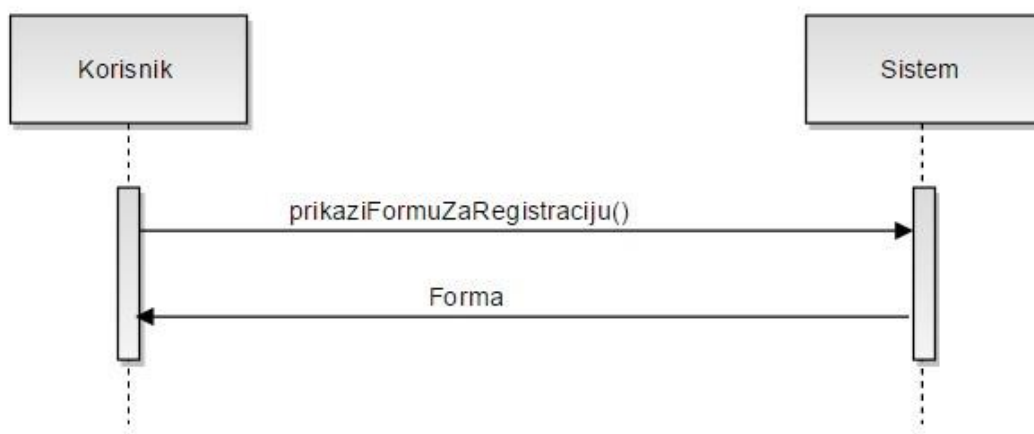
Preduslov: Sistem je uključen i prikazuje formu za registrovanje korisnika.

Osnovni scenario SK

1. Korisnik **unosí** podatke za registraciju.
2. Korisnik **poziva** sistem da sačuva podatke o korisniku.
3. Sistem **pamti** podatke o korisniku.
4. Sistem **pokazuje** korisniku poruku: "Uspešno ste se registrovali"

Alternativna scenarija

- 4.1 Ukoliko sistem ne može da sačuva korisnika, on prikazuje korisniku poruku "Korisnik nije sačuvan, pokušajte ponovo", prekida se izvršenje scenarija.
- 4.2 Ukoliko postoji korisnik registrovan sa istim korisničkim imenom sistem prikazuje poruku "Korisnik sa tim korisničkim imenom već postoji, molimo pokušajte sa nekim drugim korisničkim imenom!"



Slika 26 Registracija korisnika

2. Slučaj korišćenja - Prijavljivanje korisnika

Naziv SK

Prijavljivanje korisnika

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

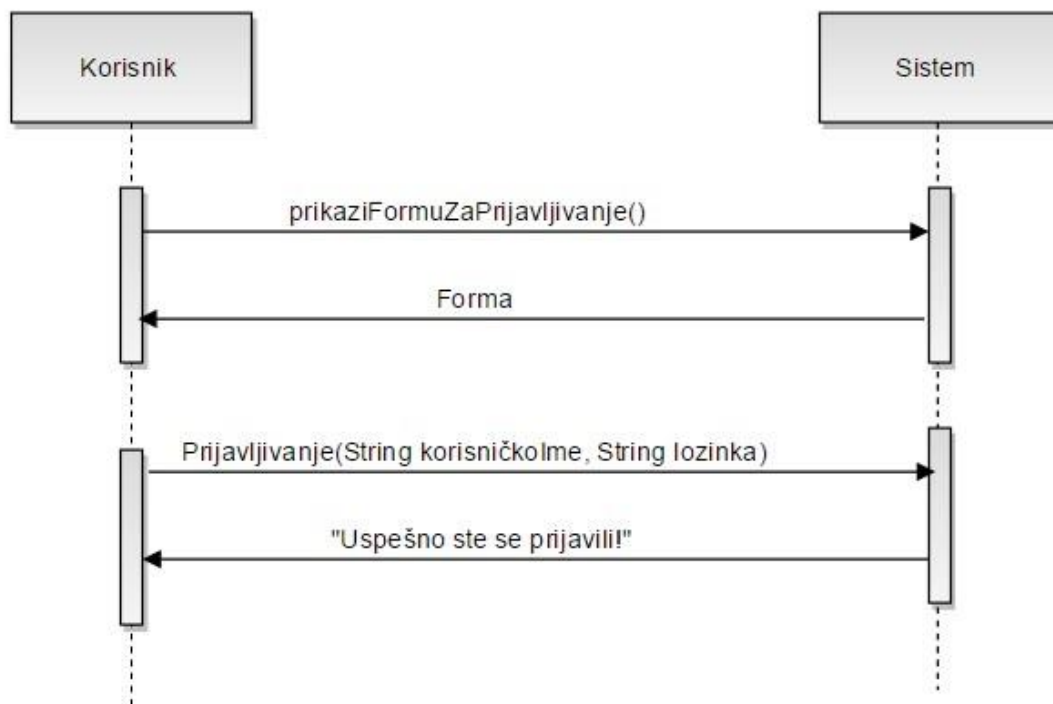
Preduslov: Sistem je uključen i prikazuje formu za prijavljivanje korisnika.

Osnovni scenario SK

1. Korisnik **unos**i podatke za prijavljivanje.
2. Korisnik **poziva** sistem da proveri podatke o korisniku.
3. Sistem **prikazuje** korisniku poruku "Uspešno ste se prijavili".

Alternativna scenarija

3.1 Ukoliko sistem ne može da pronađe korisnika, on prikazuje korisniku poruku "Neispravni podaci za prijavljivanje, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 27 Prijavljivanje korisnika

3. Slučaj korišćenja - Odjavljivanje korisnika

Naziv SK

Odjavljivanje korisnika

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

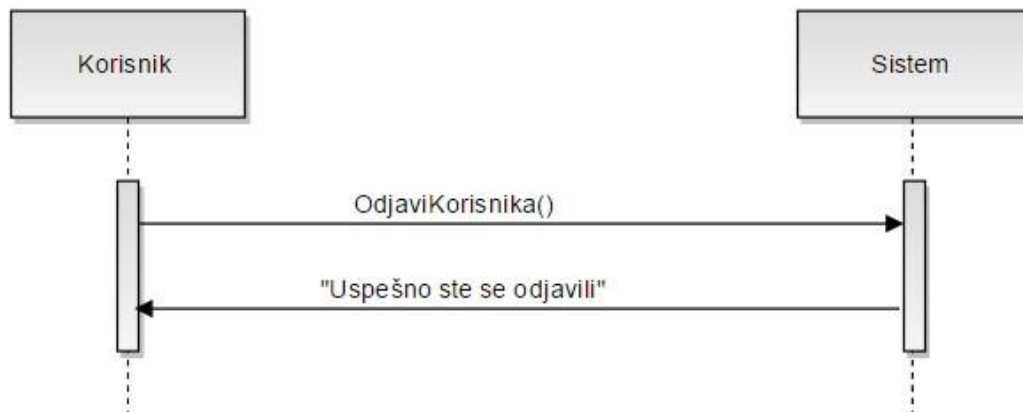
Preduslov: Sistem je uključen i korisnik je prijavljen.

Osnovni scenario SK

1. Korisnik **poziva** sistem da ga odjavi.
3. Sistem **prikazuje** korisniku poruku "Uspešno ste se odjavili".

Alternativna scenarija

2.1 Ukoliko sistem ne može da odjavi korisnika, on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 28 Odjavljivanje korisnika

4. Slučaj korišćenja - Kreiranje odeljenja

Naziv SK

Kreiranje odeljenja

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

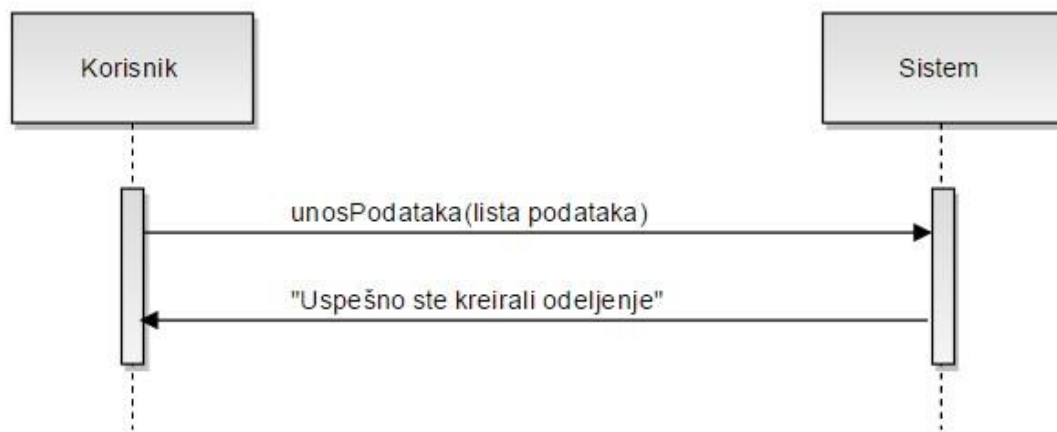
Preduslov: Sistem je uključen i korisnik je prijavljen. Sistem prikazuje formu za kreiranje odeljenja. Korisnik ima ulogu administratora.

Osnovni scenario SK

1. Korisnik **unos**i podatke o novom odeljenju.
2. Korisnik **poziva** sistem da zapamti novo odeljenje.
3. Sistem **prikazuje** korisniku poruku "Uspešno ste kreirali odeljenje".

Alternativna scenarija

2.1 Ukoliko sistem ne može da zapamti odeljenje, on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 29 Unos odeljenja

5. Slučaj korišćenja - Prikaz odeljenja

Naziv SK

Prikaz odeljenja

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

Preduslov: Sistem je uključen i korisnik je prijavljen. Korisnik ima ulogu administratora.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže stranicu sa odeljenjima.
2. Sistem **prikazuje** korisniku stranicu sa odeljenjima.

Alternativna scenarija

- 2.1 Ukoliko sistem ne može da prikaže stranicu sa odeljenjima , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 30 Prikaz odeljenja

6. Slučaj korišćenja - Kreiranje zadataka

Naziv SK

Kreiranje zadataka

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

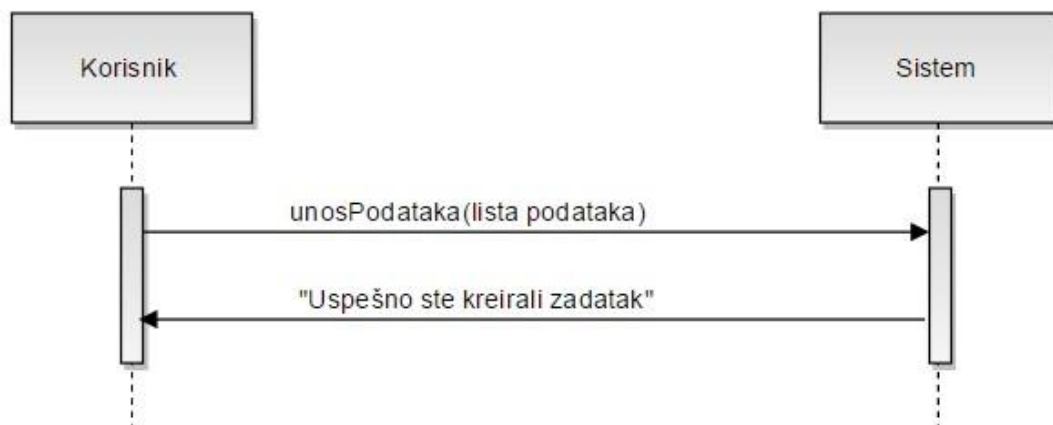
Preduslov: Sistem je uključen i korisnik je prijavljen. Sistem prikazuje formu za kreiranje zadataka. Korisnik ima ulogu administratora.

Osnovni scenario SK

1. Korisnik **unos**i podatke o novom zadatku.
2. Korisnik **poziva** sistem da zapamti novi zadatak.
3. Sistem **prikazuje** korisniku poruku "Uspešno ste kreirali zadatak".

Alternativna scenarija

3.1 Ukoliko sistem ne može da zapamti zadatak, on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 31 Unos zadatka

7. Slučaj korišćenja - Prikaz zadataka

Naziv SK

Prikaz zadataka

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

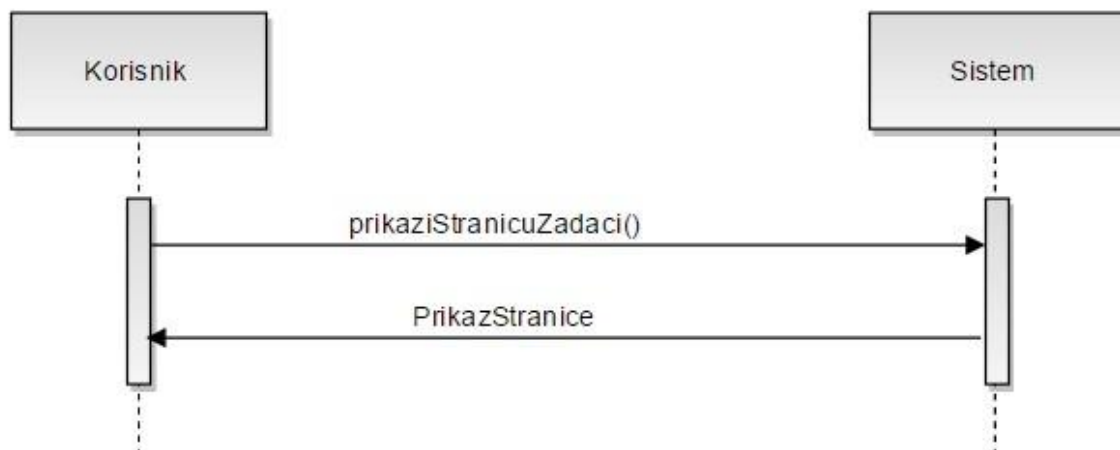
Preduslov: Sistem je uključen i korisnik je prijavljen. Korisnik ima ulogu administratora.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže stranicu sa zadacima.
2. Sistem **prikazuje** korisniku stranicu sa zadacima.

Alternativna scenarija

2.1 Ukoliko sistem ne može da prikaže stranicu sa zadacima , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 32 Prikaz zadataka

8. Slučaj korišćenja - Izmena zadataka

Naziv SK

Izmena zadataka

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

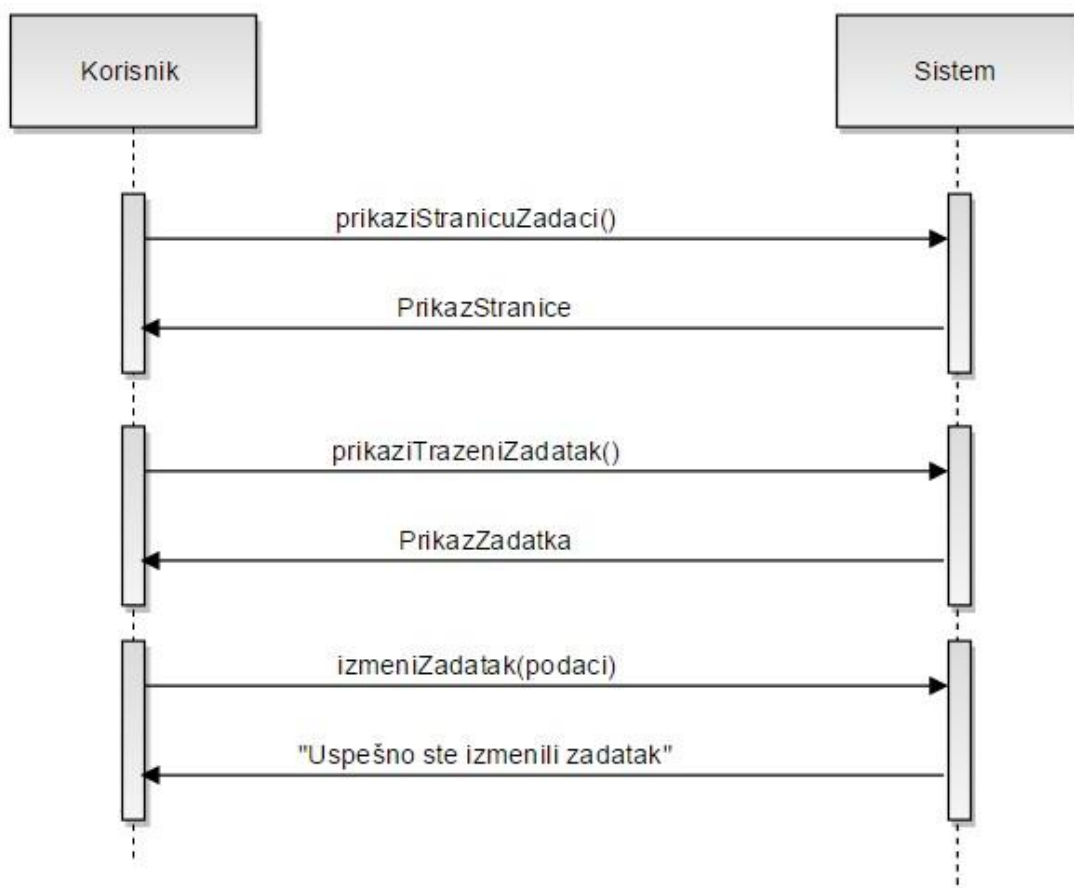
Preduslov: Sistem je uključen i korisnik je prijavljen. Korisnik ima ulogu administratora.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže stranicu sa zadacima.
2. Sistem **prikazuje** korisniku stranicu sa zadacima.
3. Korisnik **poziva** sistem da vrati zadatak koji želi da izmeni.
4. Sistem **prikazuje** traženi zadatak.
5. Korisnik **vrši** izmene na zadatku.
6. Korisnik **poziva** sistem da zapamti izmene.
7. Sistem **prikazuje** poruku "Uspešno ste izmenili zadatak".

Alternativna scenarija

- 2.1 Ukoliko sistem ne može da prikaže stranicu sa zadacima , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenaija.
- 4.1 Ukoliko sistem ne može da prikaže traženi zadatak , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenaija.
- 7.1 Ukoliko sistem ne može da sačuva promene nad zadatkom , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 33Izmena zadatka

9. Slučaj korišćenja - Brisanje zadatka

Naziv SK

Brisanje zadatka

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

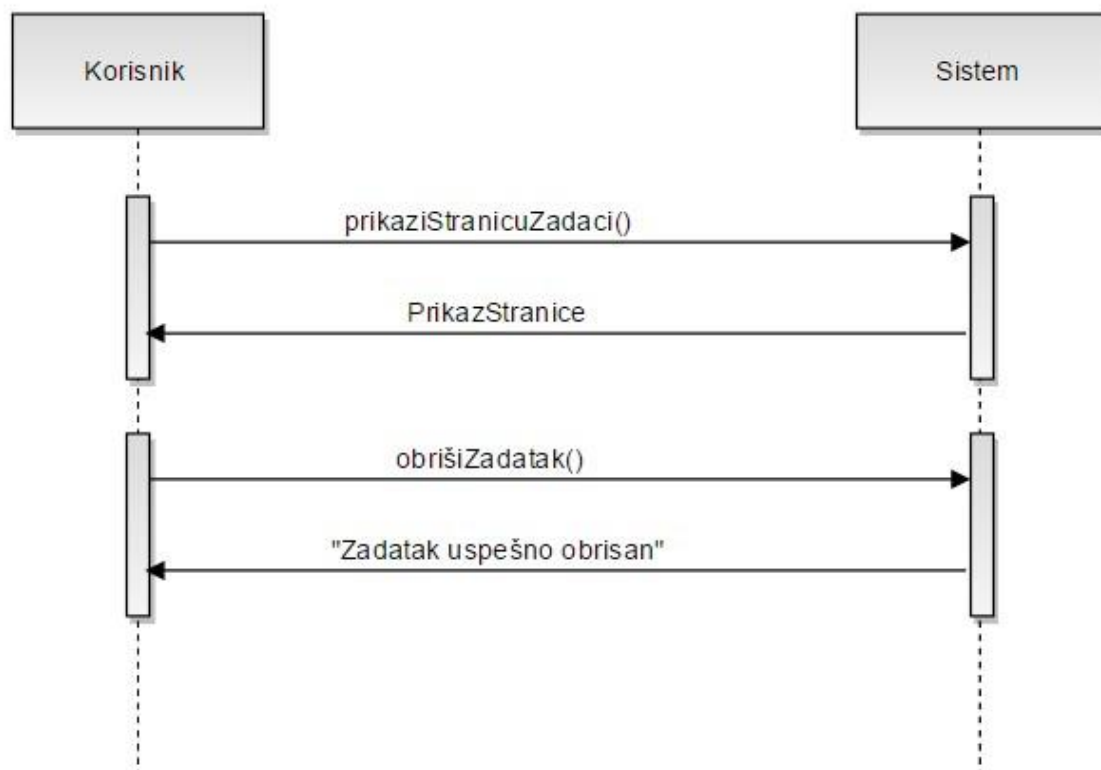
Preduslov: Sistem je uključen i korisnik je prijavljen. Korisnik ima ulogu administratora.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže stranicu sa zadacima.
2. Sistem **prikazuje** korisniku stranicu sa zadacima.
3. Korisnik **poziva** sistem da obriše zadatak.
4. Sistem **prikazuje** poruku "Uspešno ste obrisali zadatak".

Alternativna scenarija

4.1 Ukoliko sistem ne može da obriše zadatak , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 34 Brisanje zadatka

10. Slučaj korišćenja - Kreiranje članka

Naziv SK

Kreiranje članka

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

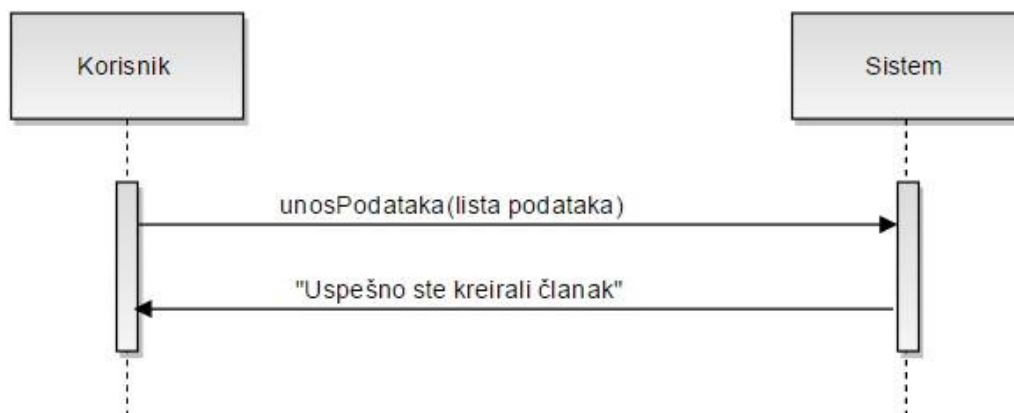
Preduslov: Sistem je uključen i korisnik je prijavljen. Sistem prikazuje formu za kreiranje članka.

Osnovni scenario SK

1. Korisnik **unos**i podatke o novom članku.
2. Korisnik **poziva** sistem da zapamti novi članak.
3. Sistem **prikazuje** korisniku poruku "Uspešno ste kreirali članak".

Alternativna scenarija

3.1 Ukoliko sistem ne može da zapamti članak, on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 35 Unos članka

11. Slučaj korišćenja - Prikaz članka

Naziv SK

Kreiranje članka

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

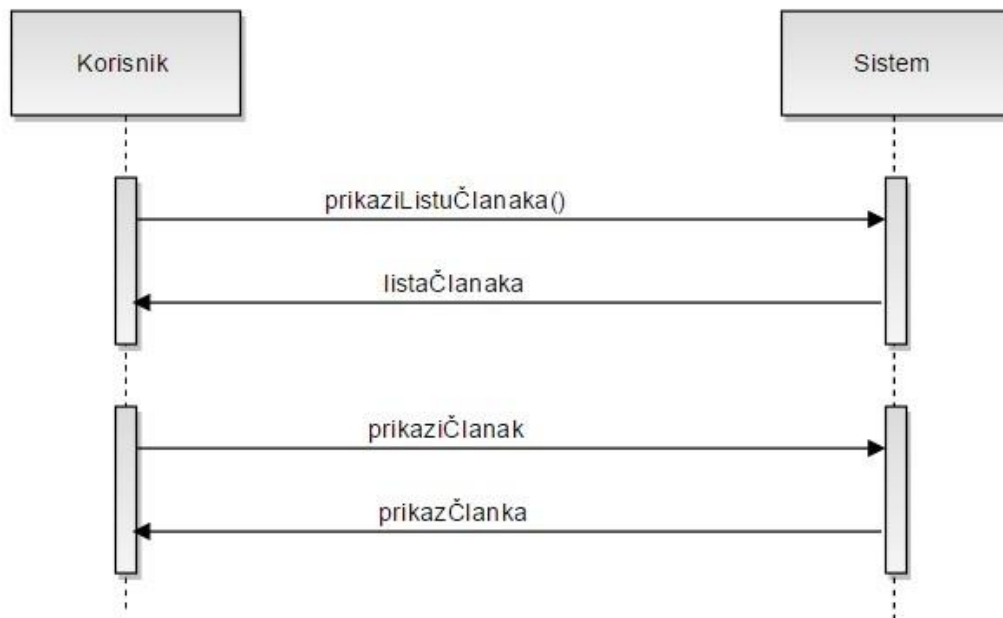
Preduslov: Sistem je uključen i korisnik je prijavljen.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže listu članaka.
2. Sistem **prikazuje** korisniku listu članaka.
3. Korisnik **bira** članak koji želi da vidi.
4. Sistem **prikazuje** korisniku željeni članak.

Alternativna scenarija

4.1 Ukoliko sistem ne može da prikaže članak, on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 36 Prikaz Članka

12. Slučaj korišćenja - Izmena članka

Naziv SK

Izmena članka

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

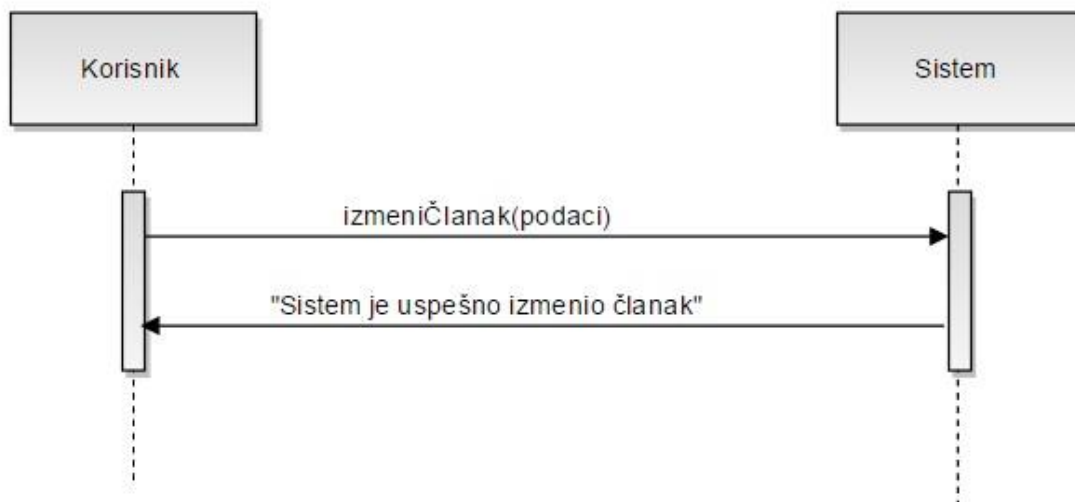
Preduslov: Sistem je uključen i korisnik je prijavljen. Korisnik je autor članka. Sistem prikazuje članak.

Osnovni scenario SK

1. Korisnik **vrši** izmenu članka.
2. Korisnik **poziva** sistem da izmeni članak.
3. Sistem **prikazuje** poruku "Uspešno ste izmenili članak".

Alternativna scenarija

3.1 Ukoliko sistem ne može da sačuva promene nad člankom , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 37 Izmena članka

13. Slučaj korišćenja - Pretraga članaka

Naziv SK

Pretraga članaka

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

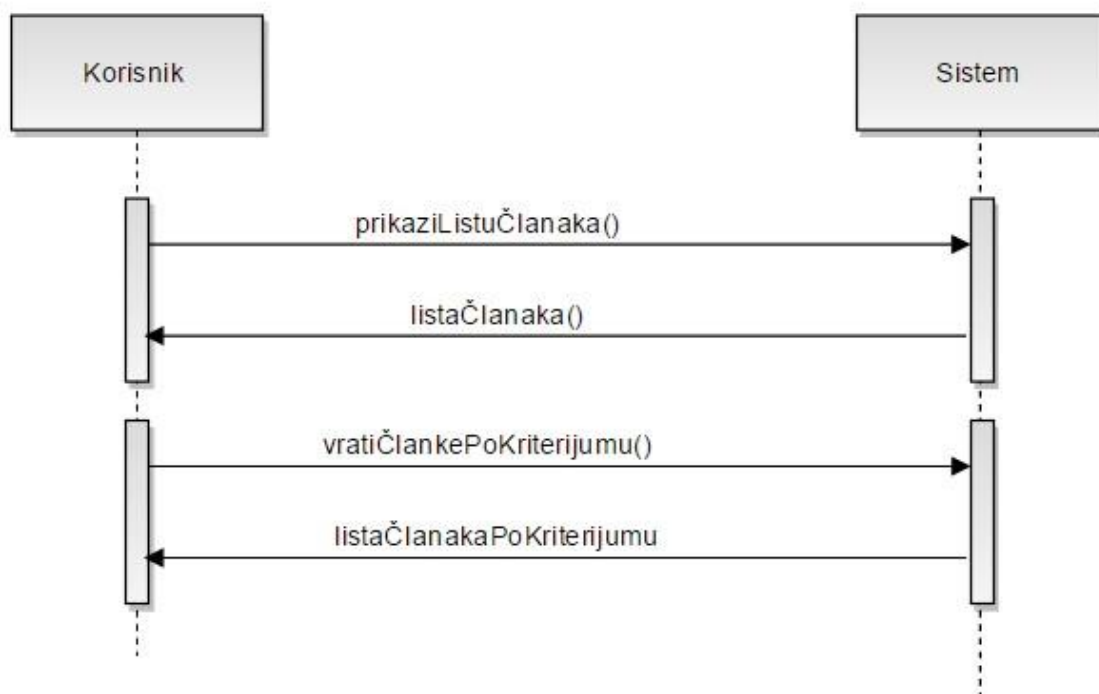
Preduslov: Sistem je uključen i korisnik je prijavljen.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže listu članaka.
2. Korisnik **unosí** parametre pretrage.
3. Korisnik **poziva** sistem da prikaže članke.
4. Sistem **prikazuje** listu članaka koji odgovaraju kriterijumu pretrage.

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe članke po zadatim kriterijumima , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 38 Pretraga Članaka

14. Slučaj korišćenja - Deljenje video sadržaja

Naziv SK

Deljenje video sadržaja

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

Preduslov: Sistem je uključen i korisnik je prijavljen.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže formu za deljenje video sadržaja.
2. Korisnik **unos**i podatke za video.
3. Korisnik **poziva** sistem da sačuva video.

4. Sistem **prikazuje** poruku "Video je uspešno sačuvan".

Alternativna scenarija

4.1 Ukoliko sistem ne može da sačuva video , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 39 Unos video sadržaja

15. Slučaj korišćenja - Prikaz video sadržaja

Naziv SK

Prikaz video sadržaja

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

Preduslov: Sistem je uključen i korisnik je prijavljen.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže listu video sadržaja.
2. Korisnik **bira** video koji želi da pogleda.
3. Korisnik **poziva** sistem da prikaže video.
4. Sistem **prikazuje** video korisniku.

Alternativna scenarija

4.1 Ukoliko sistem ne može da prikaže video , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 40 Prikaz Video sadržaja

16. Slučaj korišćenja - Deljenje linkova

Naziv SK

Deljenje linkova

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

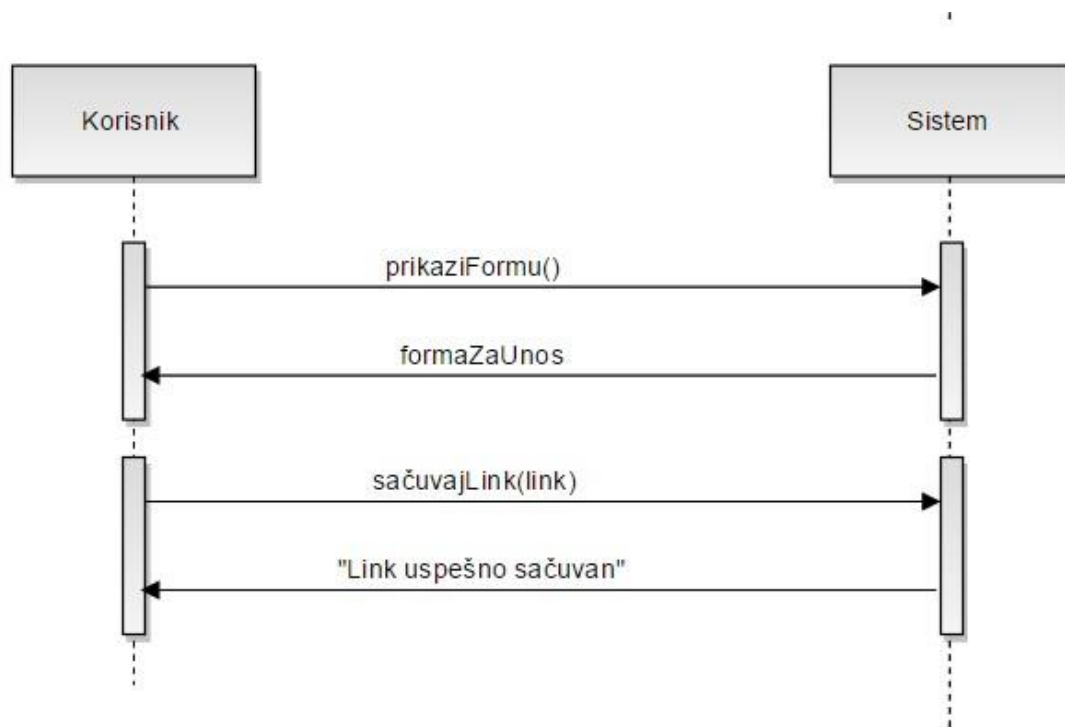
Preduslov: Sistem je uključen i korisnik je prijavljen.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže formu za deljenje linka.
2. Korisnik **unos**i link koji zeli da podeli.
3. Korisnik **poziva** sistem da zapamti link.
4. Sistem **prikazuje** poruku "Link uspešno sačuvan".

Alternativna scenarija

4.1 Ukoliko sistem ne može da sačuva link , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 41 Deljenje linka

17. Slučaj korišćenja - Prikaz Linkova

Naziv SK

Prikaz linkova

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

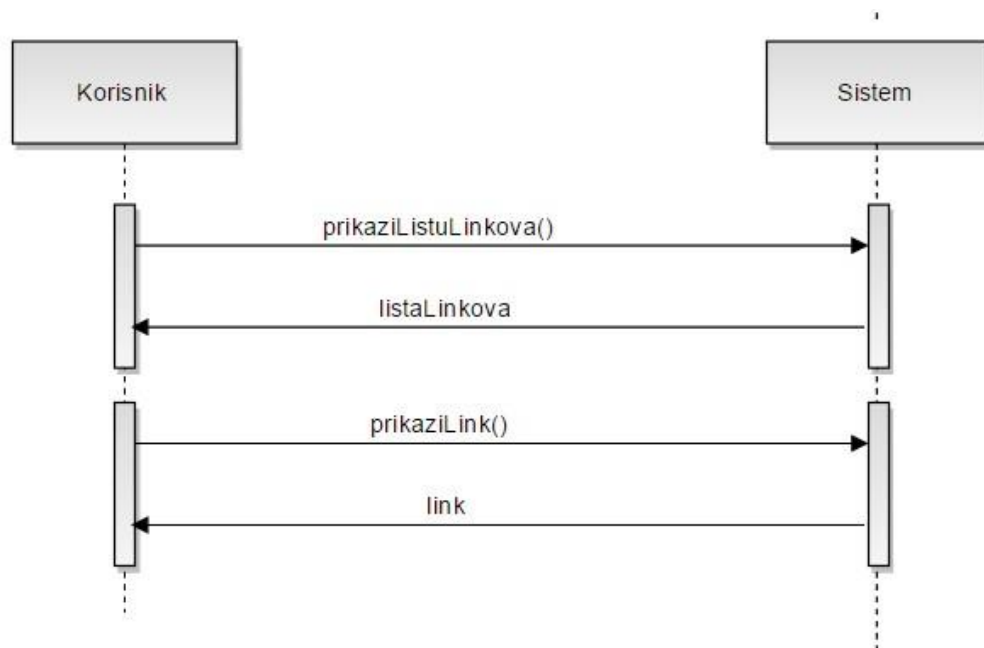
Preduslov: Sistem je uključen i korisnik je prijavljen.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže listu linkova.
2. Korisnik **bira** link koji želi da pogleda.
3. Sistem **prikazuje** traženi link.

Alternativna scenarija

3.1 Ukoliko sistem ne može da prikaže link , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 42 Prikaz linka

18. Slučaj korišćenja - Kreiranje kategorija

Naziv SK

Kreiranje kategorija

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

Preduslov: Sistem je uključen i korisnik je prijavljen.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže formu za rad sa kategorijama.
2. Korisnik **unos**i novu kategoriju.
3. Korisnik **poziva** sistem da zapamti kategoriju.
4. Sistem **prikazuje** poruku "Kategorija uspešno sačuvana".

Alternativna scenarija

- 4.1 Ukoliko sistem ne može da sačuva kategoriju , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 43 Unos kategorije

19. Slučaj korišćenja - Pretraga najčešćih pitanja

Naziv SK

Pretraga najčešćih pitanja

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

Preduslov: Sistem je uključen i korisnik je prijavljen.

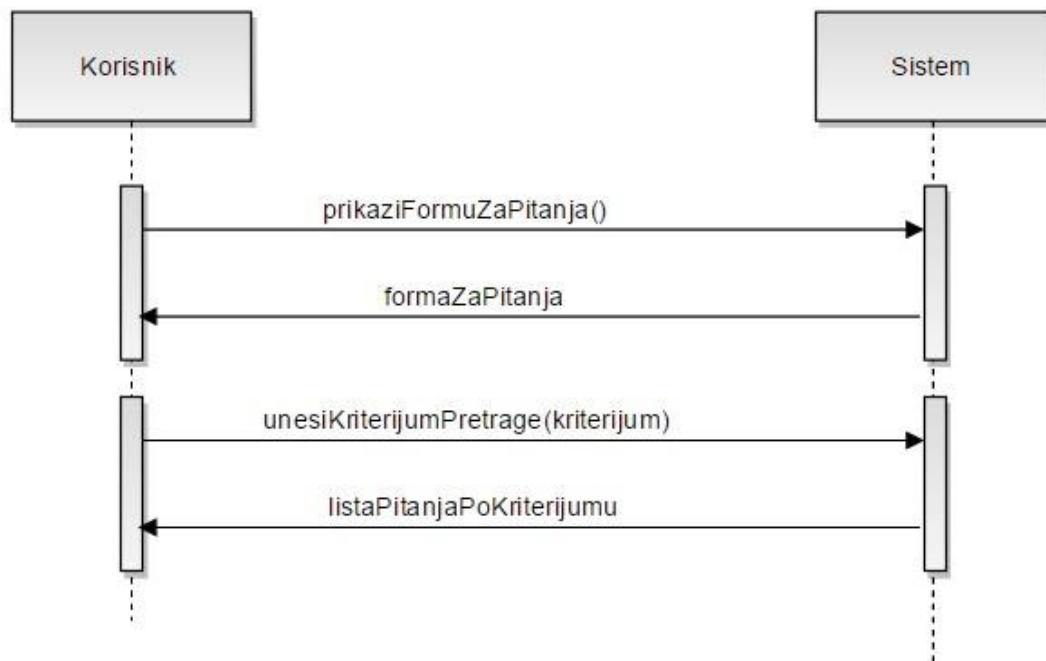
Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže formu za rad sa pitanjima.
2. Korisnik **unos**i kriterijum pretrage.
3. Korisnik **poziva** sistem da pretraži pitanja.

4. Sistem **prikazuje** korisniku pitanja po zadanom kriterijumu.

Alternativna scenarija

4.1 Ukoliko sistem ne može da pronađe pitanja , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 44 Pretraga Pitanja

20. Slučaj korišćenja - Pregled profila korisnika

Naziv SK

Pregled profila korisnika

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

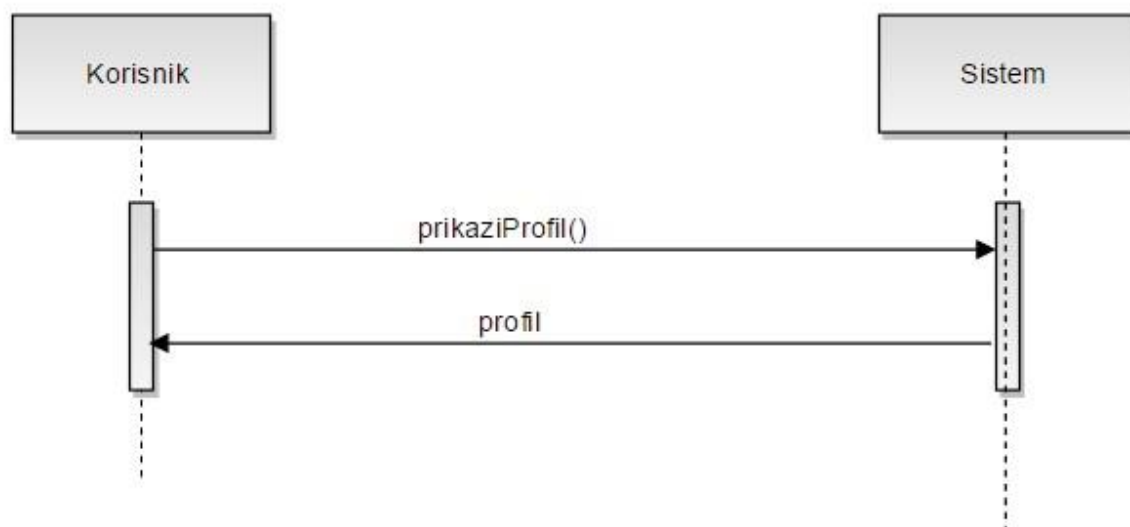
Preduslov: Sistem je uključen i korisnik je prijavljen.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže profil korisnika.
2. Sistem **prikazuje** korisniku traženi profil.

Alternativna scenarija

2.1 Ukoliko sistem ne može da pronađe profil , on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 45 Prikaz profila

21. Slučaj korišćenja - Kreiranje postova sa kodom

Naziv SK

Kreiranje postova sa kodom

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

Preduslov: Sistem je uključen i korisnik je prijavljen.

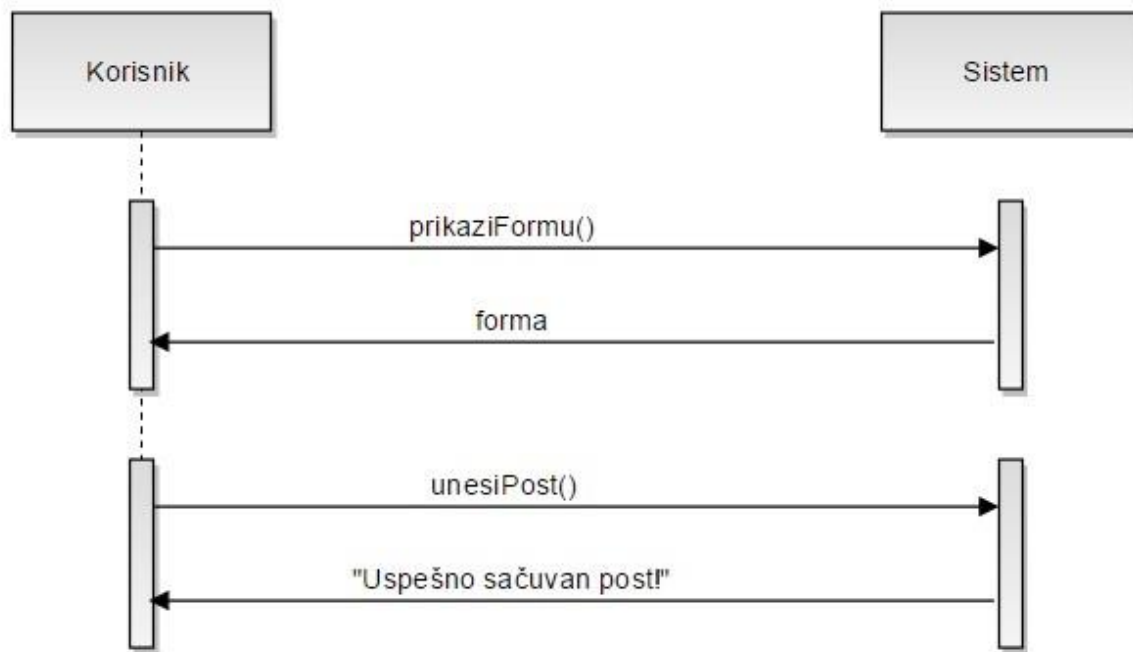
Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže formu za rad sa postovima sa kodom.

2. Korisnik **unosí** novi post sa kodom.
3. Korisnik **poziva** sistem da zapamti post sa kodom.
4. Sistem **prikazuje** poruku " Post sa kodom uspešno sačuvan".

Alternativna scenarija

4.1 Ukoliko sistem ne može da sačuva post sa kodom, on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 46 Čuvanje posta sa kodom

22. Slučaj korišćenja - Komentarisanje entiteta

Naziv SK

Komentarisanje entiteta

Aktori SK

Korisnik

Učesnici SK

Korisnik i sistem (aplikacija)

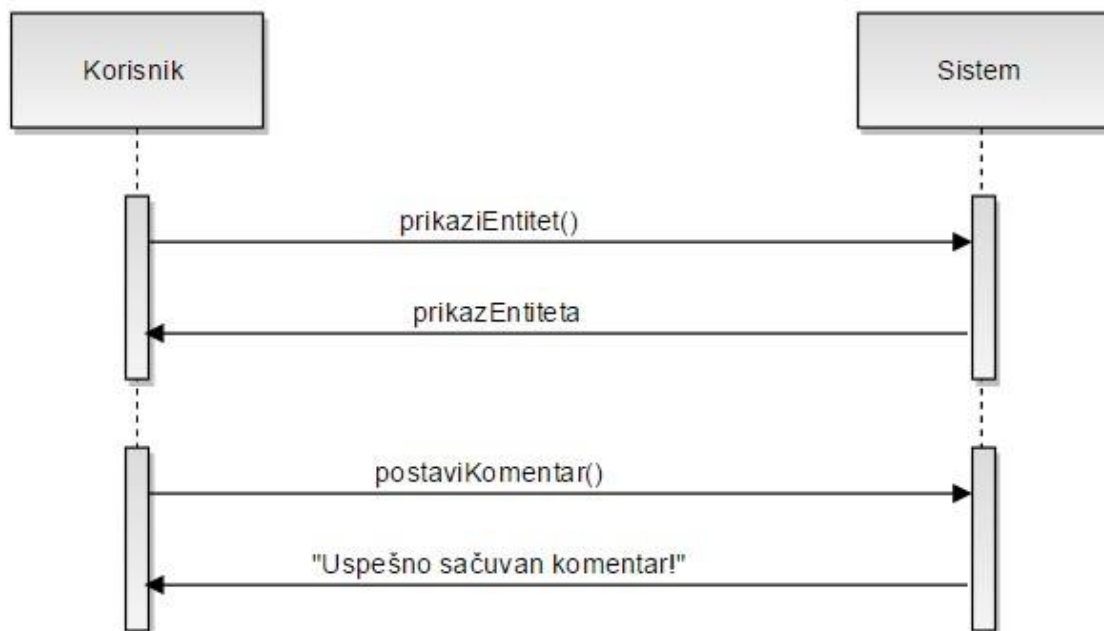
Preduslov: Sistem je uključen i korisnik je prijavljen.

Osnovni scenario SK

1. Korisnik **poziva** sistem da prikaže entitet.
2. Korisnik **unosí** novi komentar.
3. Korisnik **poziva** sistem da zapamti komentar.
4. Sistem **prikazuje** poruku " Komentar uspešno sačuvan".

Alternativna scenarija

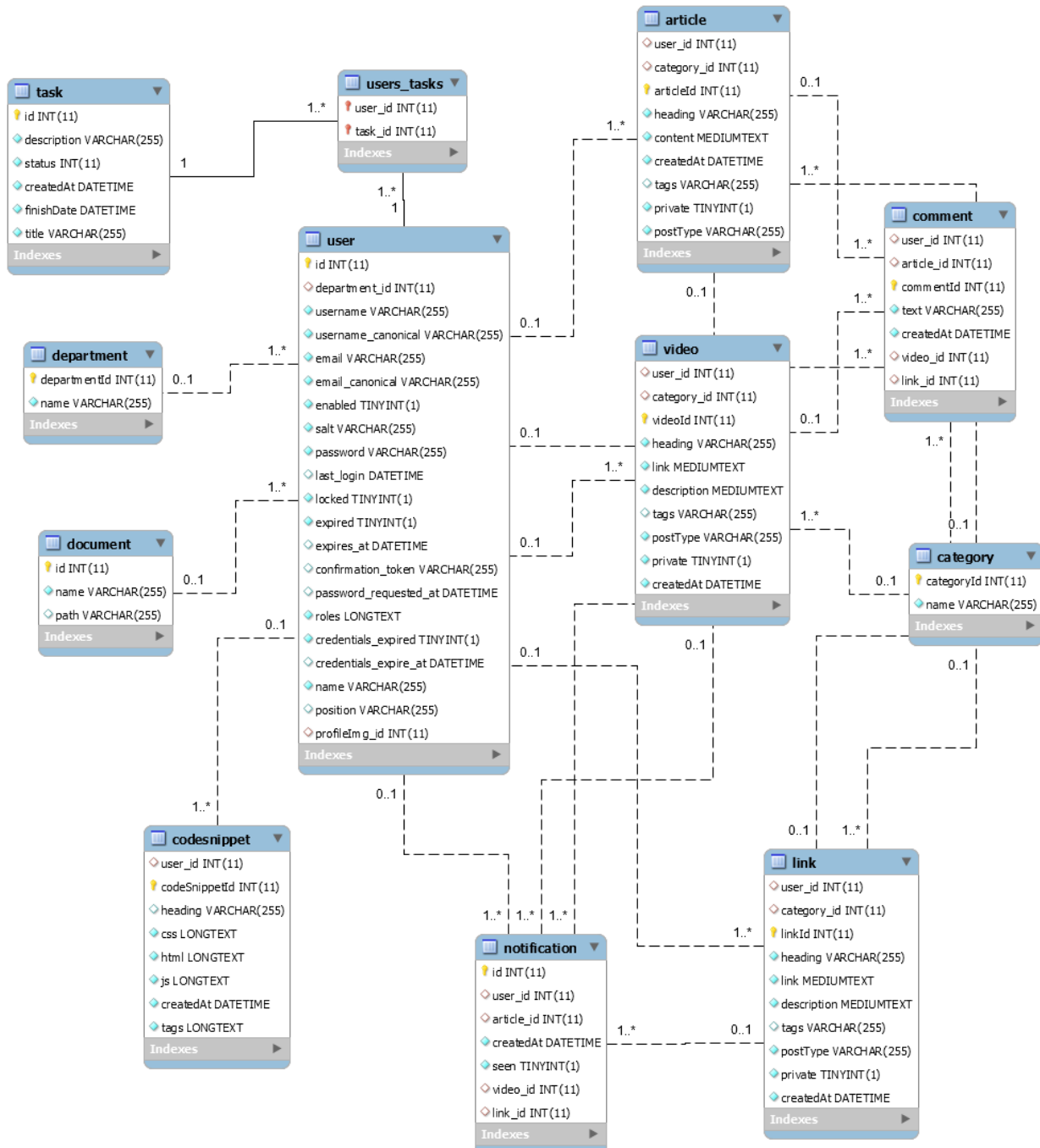
4.1 Ukoliko sistem ne može da sačuva komentar, on prikazuje korisniku poruku "Došlo je do greške, pokušajte ponovo", prekida se izvršenje scenarija.



Slika 47 Čuvanje komentara

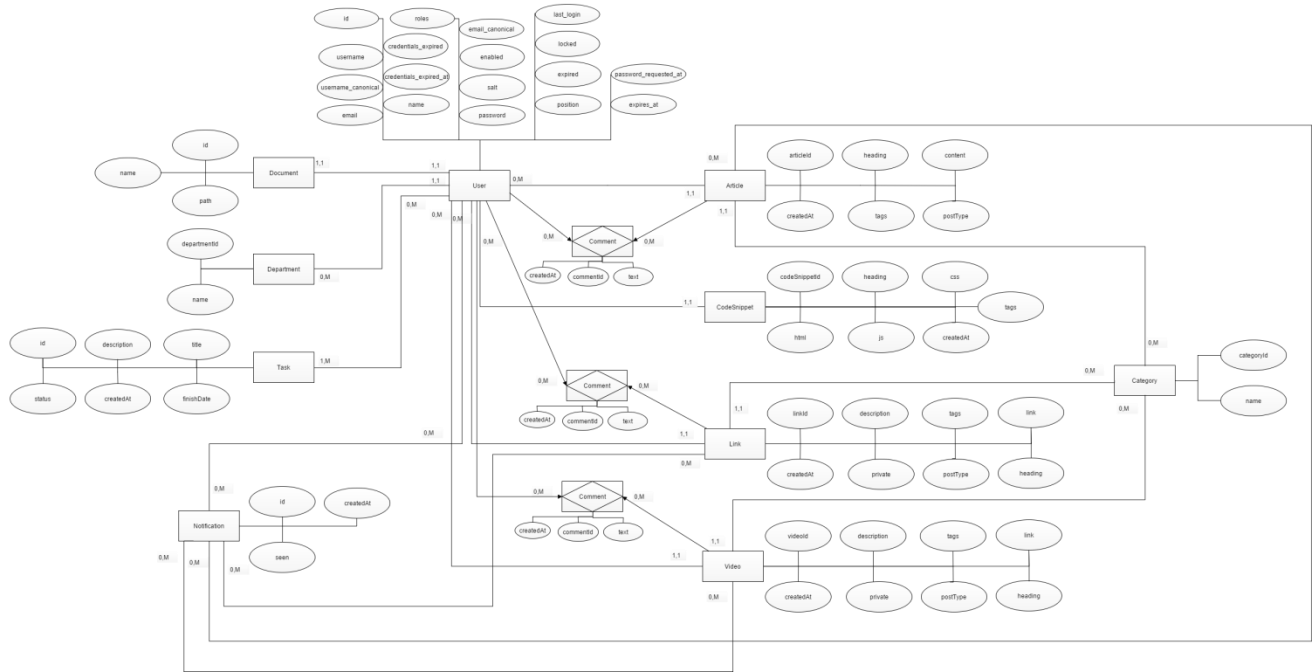
4. Analiza

1. Diagram klasa



Slika 48 Diagram klasa

2. PMOV



Slika 49 PMOV

3. Relacioni model

Na osnovu PMOV-a može se dobiti sledeći relacioni model:

USER(id, username, username_canonical, email, email_canonical, enabled, salt, password, last_login, locked, expired, expires_at, confirmation_token, password_requested_at, roles, credentials_expired, credentials_expired_at, name, position, profileImg_id, department_id)

Article(articleId, heading, content, createdAt, tags, private, postType, category_id, user_id)

Category(categoryId, name)

CodeSnippet(codeSnippetId, heading, css, html, js, createdAt, tags, user_id)

Comment(commentId, text, createdAt, video_id, article_id, link_id, user_id)

Department(departmentId, name)

Document(id, name, path)

Link(linkId, heading, link, description, tags, postType, private, createdAt, category_id, user_id)

Notification(id, createdAt, seen, user_id, video_id, link_id, article_id)

Task(id, description, status, createdAt, finishDate, title)

user_task(user_id, task_id)

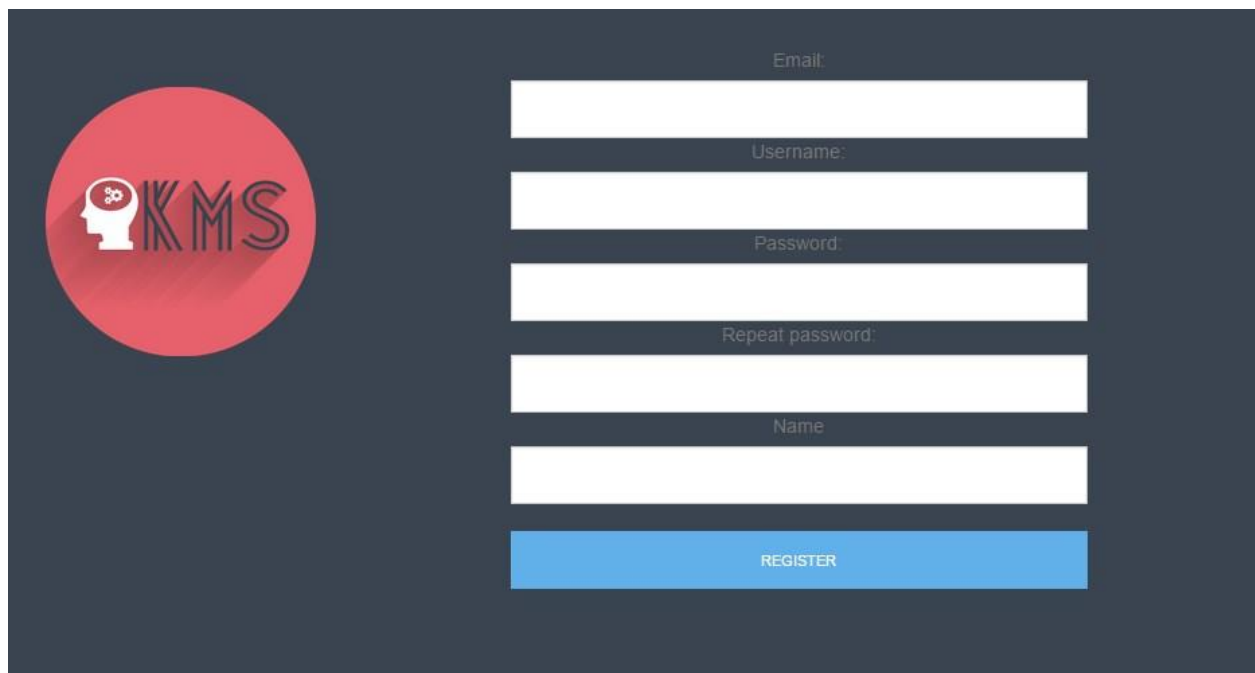
Video(videoId, heading, link, description, tags, prostType, private, createdAt, category_id, user_id)

Na osnovu relacionog modela projektovana je baza podataka. U ovom projektu korišćena je MySql relaciona baza podataka.

5. Korisničko uputstvo

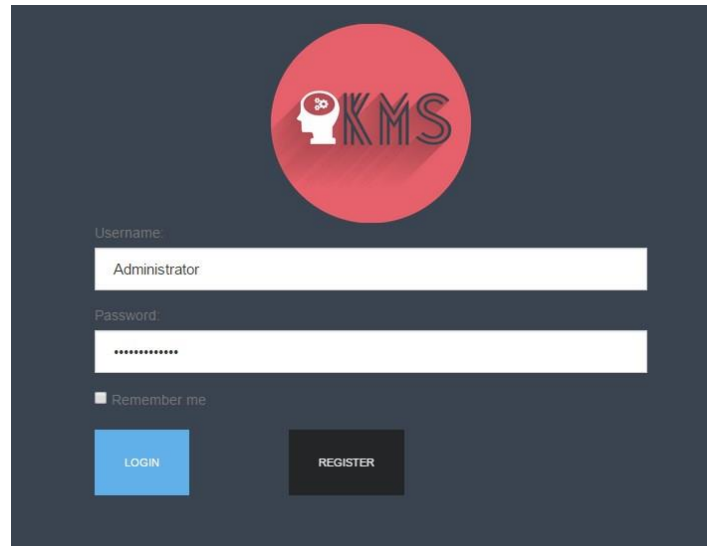
1. Prijavljivanje / Registracija

Prilikom prvog prijavljivanja na aplikaciju korisnik treba da se registruje. Prilikom registracije korisnik unosi korisničko ime, lozinku, i ime. Nakon popunjavanja forme za registraciju korisnik se registruje na aplikaciju i biva prosleđen na početnu stranicu.

The image shows a registration form for an application named KMS. On the left, there is a red circular logo with a white head icon containing a gear and the letters 'KMS'. To the right of the logo, there are six white input fields stacked vertically. The labels for these fields are 'Email:', 'Username:', 'Password:', 'Repeat password:', 'Name', and a blue 'REGISTER' button at the bottom.

Slika 50 Forma za registraciju

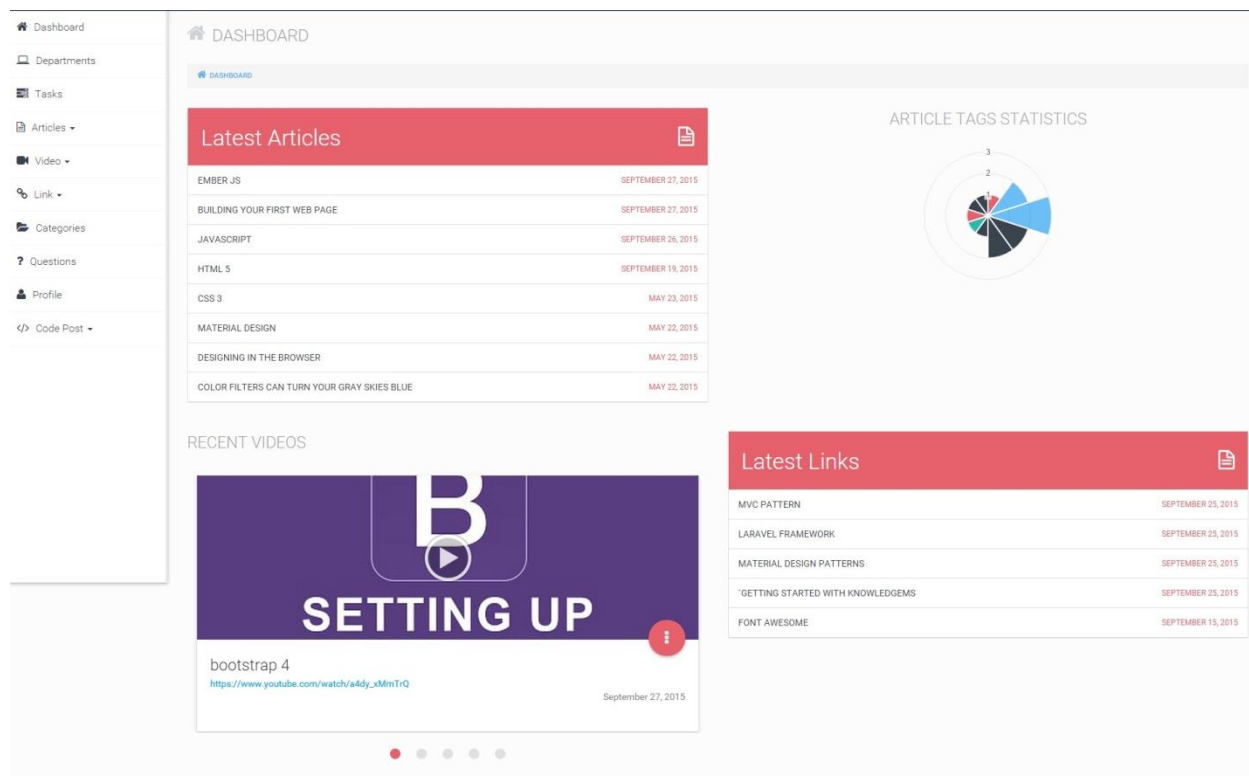
Ukoliko korisnik već poseduje nalog za aplikaciju dovoljno je da na formi za prijavljivanje popuni korisničko ime i lozinku. Ukoliko su podaci ispravno uneti sistem prijavljuje korisnika i preusmerava ga na početnu stranicu.

The image shows a login and registration interface for a system called KMS. At the top center is a red circular logo containing a white icon of a head with a gear inside, followed by the letters 'KMS' in a stylized font. Below the logo, there are two input fields: 'Username:' with the text 'Administrator' and 'Password:' with a masked password '*****'. Below the password field is a checkbox labeled 'Remember me'. At the bottom, there are two buttons: a blue 'LOGIN' button and a black 'REGISTER' button.

Slika 51 Prijavljivanje

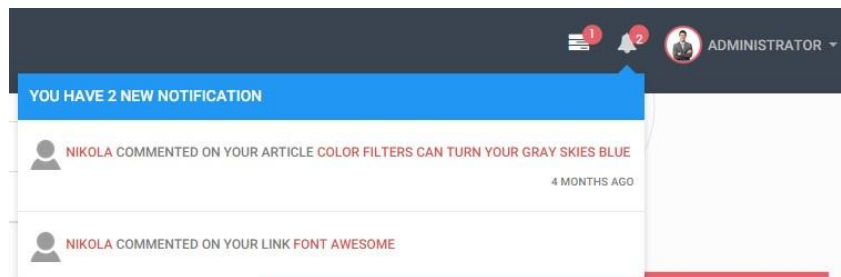
2. Početna stranica

Nakon prijavljivanja na sistem korisniku se prikazuje početna stranica. Ova stranica sadrži bočni meni za navigaciju kao i najnovije objavljene informacije. Među najnovijim informacijama se nalaze najnoviji članci, linkovi, video klipovi kao i statistika tagova koja prikazuje korisniku tagove koji su korišćeni kako bi mogao da vidi koje su teme najpopularnije.

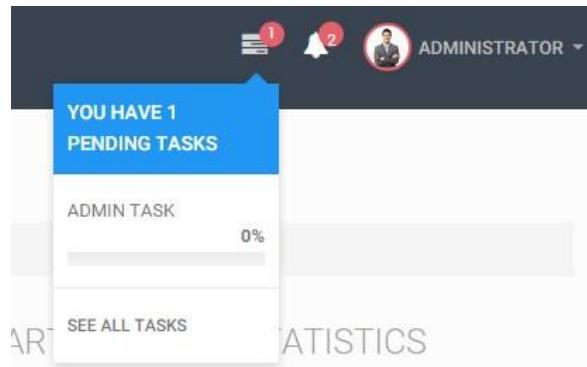


Slika 52 Početna stranica

Pored ovih informacija korisniku se u gornjem desnom uglu svake stranice, pa tako i početne, prikazuje njegova profilna slika, ime, ikonica za zadatke, ikonica za notifikacije kao i padajući meni. Padajući meni sadrži opciju odjavljivanja i link ka korisnikovom profilu. Ikonica za zadatke prikazuje trenutno aktivne zadatke, ukoliko ih ima pojaviće se broj sa brojem aktivnih zadataka. Ikonica za notifikacije prikazuje obaveštenja ukoliko je neki korisnik komentarisao članke, video klipove ili linkove koje je trenutni korisnik podelio.



Slika 53 Notifikacije



Slika 54 Zadaci

3. Odeljenja

Ovoj stranici pristup imaju samo administratori sistema. Ukoliko korisnik nije administrator ova stranica se neće pokazati u bočnom meniju a ukoliko pokuša da se pristupi preko rute pristup će mu biti odbijen. Na ovoj stranici prikazuje se forma za unos novog odeljenja kao i trenutna uneta odeljenja sa svim korisnicima koji im pripadaju. Takođe imamo i informacije o tome koliko ima odeljenja koliko ukupno imamo korisnika kao i koliko korisnika nije raspoređeno ni u jedno od odeljenja. Administrator ima mogućnost kreiranja novog odeljenja i prilikom toga bira neke od korisnika koji su neraspoređeni i smešta ih u novo odeljenje.

DEPARTMENT

DEPARTMENT / + ADD NEW

Department name

Choose users for department (hold ctrl for multiple select)

Administrator
Nikola Ugrinovic
Nikola Ugrinovic

ADD DEPARTMENT

DEPARTMENTS5

EMPLOYEES TOTAL5

UNASSIGNED EMPLOYEES3

Frontend

Id	Name	Position	Email
3	Pavle Losic	Frontend Developer	pavlelosic@yahoo.com

Backend

Id	Name	Position	Email
4	Peter Thomas		peter@gmail.com

Slika 55 Kreiranje odeljenja

4. Zadaci

Stranici sa zadacima takođe pristup imaju samo administratori aplikacije. Na ovoj stranici se vrši unos, izmena kao i brisanje zadataka. Takođe postoji prikaz trenutnih zadataka, korisnika koji učestvuju na tom zadatku kao i neke dodatne informacije koje zadatak sadrži.

TASKS

TASKS

+Add new task

Task Title

Date From

Date To

Select Employees

Administrator
Nikola Ugrinovic
Pavle Losic
Peter Thomas
Nikola Ugrinovic

Description

SAVE

Task	Date from	Date to	Assignee	Status	Actions
Register Form design	April 08, 2015	February 09, 2015	Nikola Ugrinovic Pavle Losic		
Admin task	August 31, 2015	August 15, 2015	Administrator		
Task	September 10, 2015	September 12, 2015	Nikola Ugrinovic		

+

Slika 56 Stranica za rad sa zadacima

Klikom na ikonicu plus u donjem desnom uglu otvara se forma za dodavanje zadatka. Ispod forme se nalaze izlistani zadaci, u poslednjoj koloni tabele sa zadacima nalaze se dve ikonice jedna za izmenu druga za brisanje zadatka. Ukoliko se izabere izmena zadatka otvara se forma sa popunjenim podacima i korisnik može da ih izmeni i sačuva. Ukoliko korisnik želi da obriše zadatak klikom na drugu ikonicu nakon čega se pojavljuje dijalog koji proverava da li je korisnik siguran da želi da obriše zadatak.

5. Kategorije

Unos članaka, linkova i video klipova zahteva da se oni svrstaju u neku od kategorija. Da bi se to omogućilo neophodno je da se prvo te kategorije unesu na ovoj stranici.

CATEGORY

CATEGORY / + ADD

Category name

SAVE

Slika 57 Kategorije

6. Članci

Korisnik može izlistati članke kao i filtrirati ih po kategorijama radi lakšeg pronalaska traženog članka. Klikom na neki od tih članaka korisnik se preusmerava na stranicu sa tim člankom. Na toj stranici prikazuje se sadržaj članka, komentari na članak i neke dodatne informacije vezane za članak. Pored pregleda članka korisnik može dodati članak. Ovoj stranici se pristupa kada se iz bočnog menija izabere opcija članak nakon čega se otvara padajući meni sa opcijom da se doda novi članak.

ARTICLES

ARTICLES / + ADD NEW

Insert Code Snippets

CSS PHP JS

Heading

Normal **B** U *I*

Choose category

Web design

Tags

Choose privacy

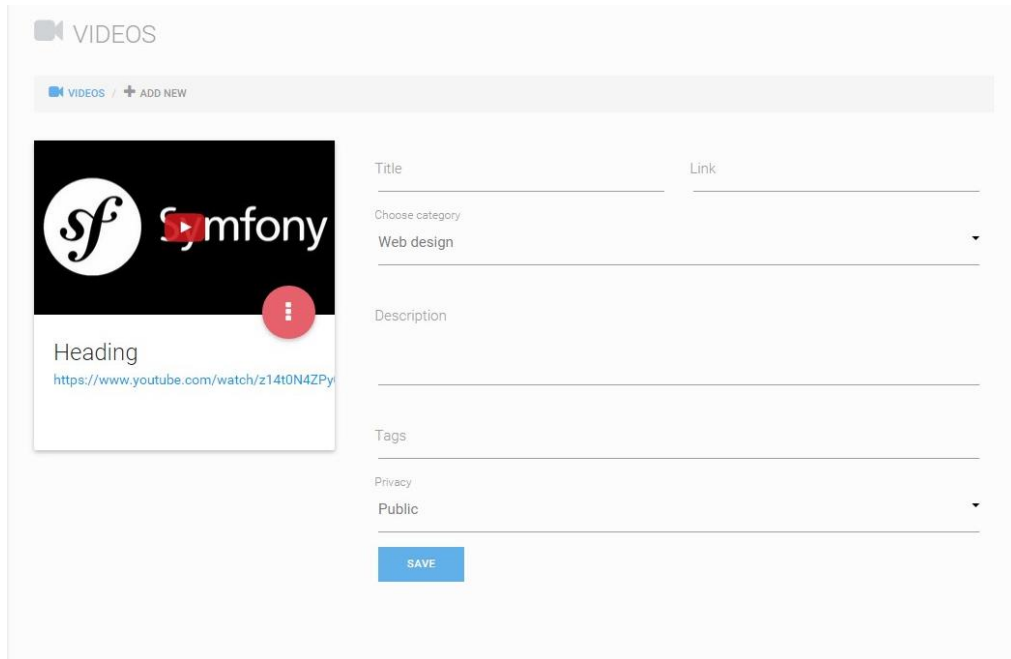
Public

ADD ARTICLE

Slika 58 Dodavanje članka

7. Video sadržaj

Korisnici takođe imaju mogućnost deljenja i izlistavanja video sadržaja. Izlistani video sadržaji se mogu pretraživati po naslovu. Deljenje video sadržaja je omogućeno pomoću Youtube sajta. Na stranici za dodavanje korisnik unosi naslov, link, bira kategoriju, unosi opis video sadržaja i tagove koji su povezani sa tim video sadržajem.



The screenshot shows a web interface for adding a new video. At the top, there's a header with a video camera icon and the word 'VIDEOS'. Below it, a navigation bar contains 'VIDEOS' and '+ ADD NEW'. The main form is divided into two columns. The left column features a video thumbnail placeholder with a 'sf symfony' logo, a red play button icon, and a red menu icon. Below the thumbnail, there's a 'Heading' label and a text input field containing a YouTube URL. The right column contains several form fields: 'Title' and 'Link' (both with text input fields), 'Choose category' (a dropdown menu currently showing 'Web design'), 'Description' (a text input field), 'Tags' (a text input field), 'Privacy' (a dropdown menu currently showing 'Public'), and a blue 'SAVE' button at the bottom.

Slika 59 Dodavanje video sadržaja

8. Linkovi

Kao i video sadržaj korisnik može izlistati podeljene linkove kao i podeliti nove sa ostalim korisnicima. Prilikom izlistavanja linkova ukoliko korisnik pređe kursorom preko linka otvoriće mu se umanjeni prikaz sadržaja koji se nalazi na tom linku. Klikom na ikonicu za više informacija korisniku se prikazuje više informacija o linku, njegov opis i tagovi. Dodavanje linka se vrši slično kao i video sadržaja. Korisnik unosi naslov, link, opis linka, bira kategoriju i unosi tagove.

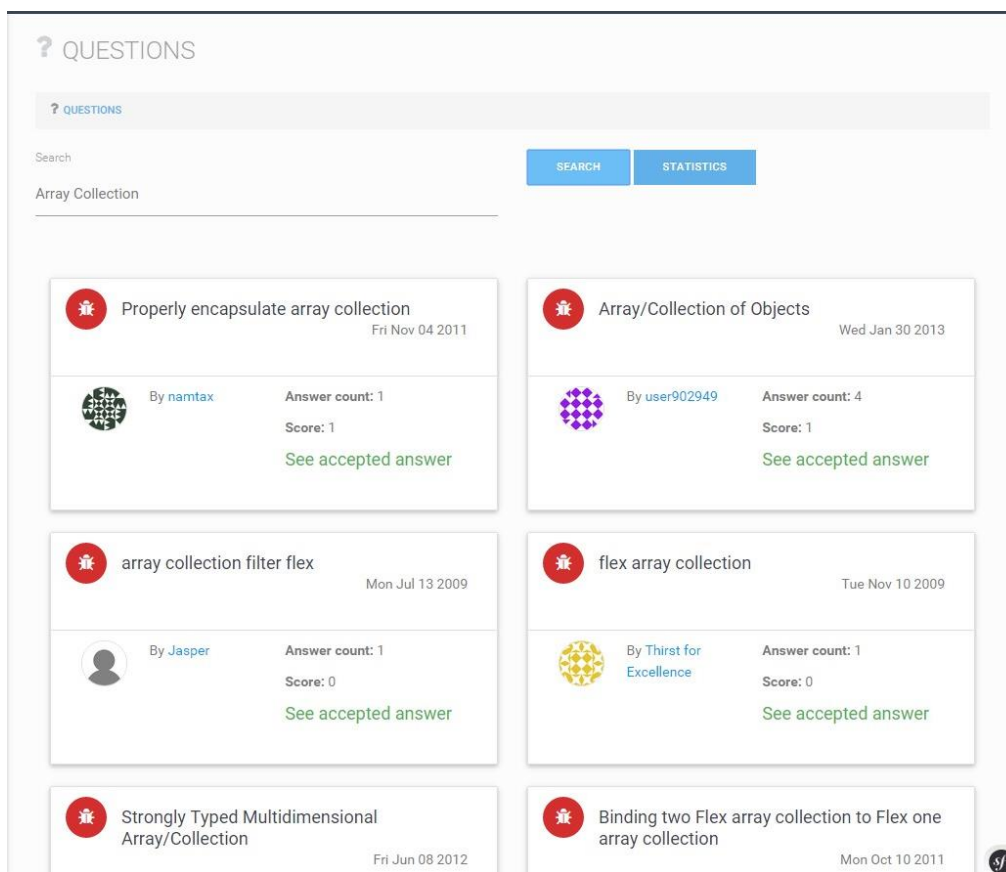
The screenshot shows a web application titled 'LINKS' with a logo of two interlocking gears. Below the title is a navigation bar with 'LINKS' and '+ ADD NEW'. The main form contains the following fields:

- Title**: A text input field.
- Link**: A text input field.
- Choose category**: A dropdown menu with 'Web design' selected.
- Description**: A text input field.
- Tags**: A text input field.
- Privacy**: A dropdown menu with 'Public' selected.
- SAVE**: A blue button at the bottom of the form.

Slika 60 Dodavanje linkova

9. Pitanja

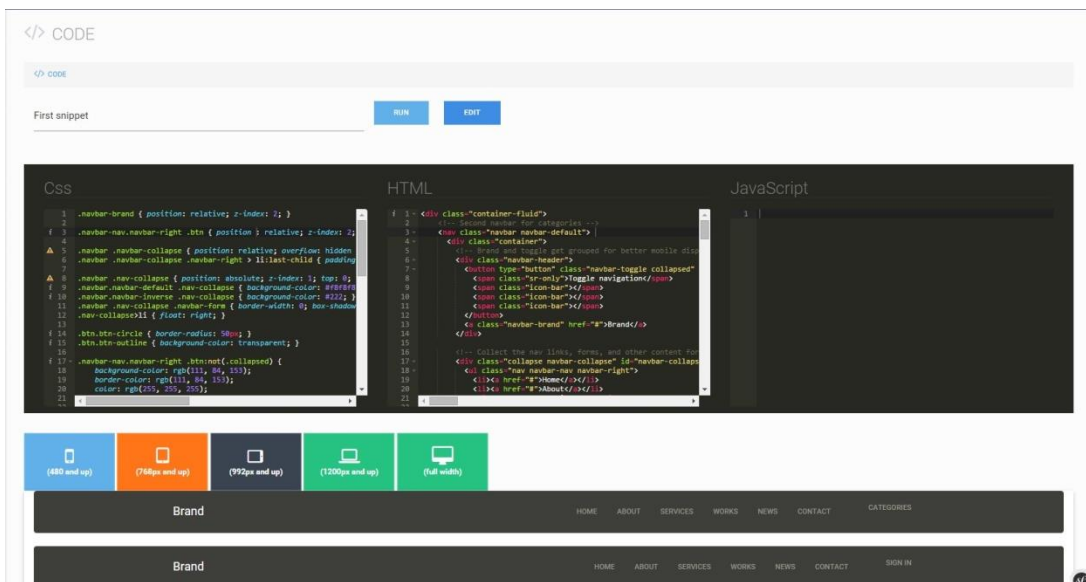
Ova stranica korisni se za pretragu najčešćih pitanja korisnika. Pitanja se pretražuju po naslovu. Prilikom ove pretrage se koristi API Stackoverflow popularnog sajta. Na zahtev se dobijaju odgovori koji odgovaraju na traženi pojam. Klikom na dugme pretraga korisniku se prikazuju odgovori a klikom na dugme statistika prikazuje se statistika tih odgovora. Svaki od odgovora sadrži neke dodatne informacije o konkretnom pitanju kao i link ka tačnom odgovoru kako bi se korisniku olakšao pronalazak rešenja.



Slika 61 Pretraga pitanja

10. Postovi sa kodom

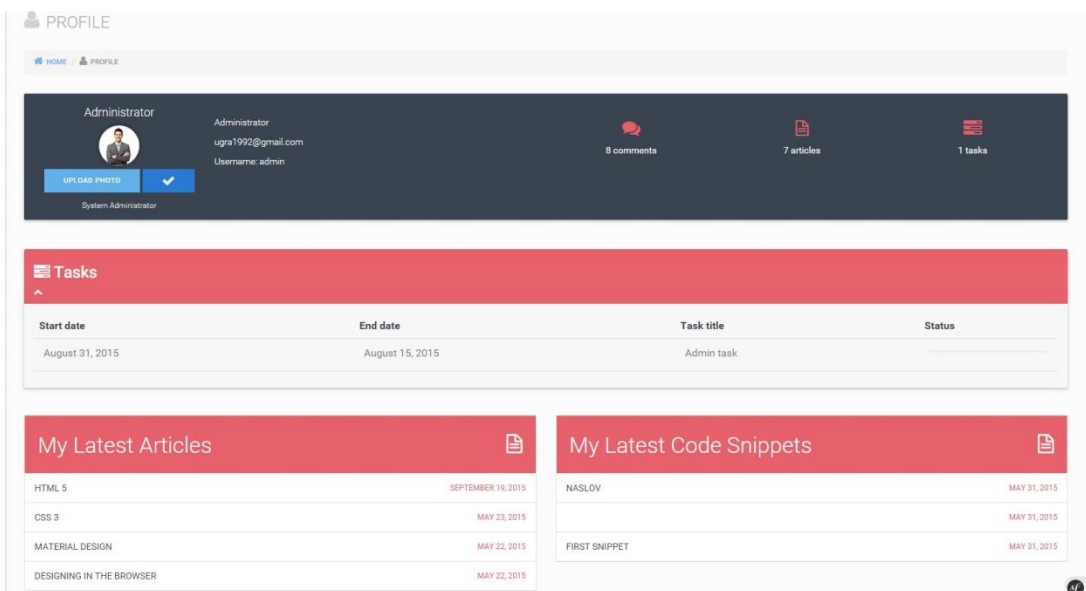
Pored deljenja standardnih članaka korisnik može kreirati postove sa kodom. Ovi postovi su specifični po tome što korisnik može unositi HTML, CSS i Javascript kod. Na stranici za dodavanje ovih postova korisniku se prikazuju tri polja u koja korisnik unosi kod. Ova polja su specijalizovana za unos tačno tih kodova. To znači da vrše automatsko popunjavanje koda kao i bojenje ključnih reči i na taj način omogućava lakše pisanje koda. Ono što dodatno olakšava kreiranje ovih postova je automatski prikaz prilikom svake izmene koda. Korisnik može birati nekoliko dimenzija u kojima želi da vidi prikaz njegovog trenutnog koda.



Slika 62 Postovi sa kodom

11. Profil

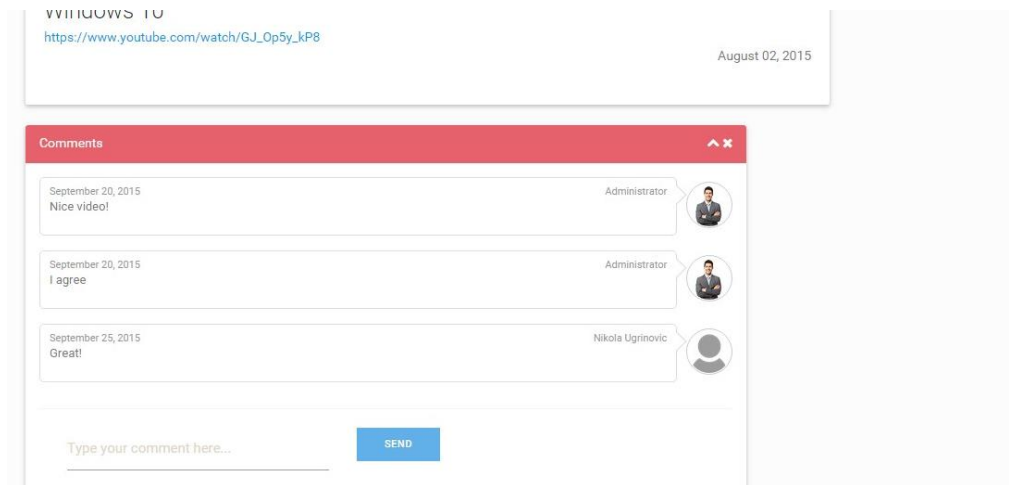
Svaki korisnik poseduje profil stranicu. Na ovoj stranici se prikazuju osnovne informacije o korisniku kao i njegove najnovije objave. Na ovoj stranici korisnik može postaviti svoju profilnu sliku koja će se prikazivati svaki put uz njegov profil.



Slika 63 Profil

12. Komentari

Korisnici mogu komentarisati skoro sve vidove objava. Korisnik poseti stranicu sadržaja koji želi da komentariše i ispod sadržaja nalazi se deo sa komentarima. Korisnik unosi komentar i nakon klika na dugme Pošalji njegov komentar se ispisuje a korisniku koji je autor tog članka se prikazuje obaveštenje u gornjem meniju.



Slika 64 Komentari

6. Zaključak

Za razvoj ove aplikacije korišćen je Symfony prvenstveno zbog velike podrške od strane njegove zajednice kao i to što je jedan od najpopularnijih php razvojnih okvira. Omogućava veliku fleksibilnost pri izradi aplikacije i samim tim je bio najbolje rešenje za izradu ove aplikacije. Istovremeno pruža veliku sigurnost aplikacije što je jako bitno. Bootstrap je korišćen da bi se omogućila responzivnost kao i da bi se olakšalo i ubrzalo pisanje CSS koda. Takođe Materialize biblioteka je dodatno uticala na izgled aplikacije. Pomoću ove biblioteke delovi aplikacije su lepo struktuirani, pregledni i laki za korišćenje. Korišćena je i Chart.js biblioteka koja je ulepšala prikaz nekih statističkih podataka.

KnowledgeMS aplikacija olakšava razmenu informacija unutar organizacije tako što različite tipove znanja i informacija grupiše na jednom mestu. Na ovaj način je pristup informacijama brži i lakši. Mogućnost pretraživanja po kategorijama i mogućnost postavljanja tagova na sve vrste materijala pomaže korisnicima da lakše dođu do željenog sadržaja.

Postoji veliki broj mogućnosti da se ova aplikacija proširi. Neke od tih mogućnosti bi bile da se uvede deljenje video sadržaja van Youtube sajta, da se uvede rangiranje sadržaja na osnovu glasova korisnika, da se vrši pretraga sadržaja na osnovu tagova ili da se korisniku predlažu članci na osnovu tema koje on izabere.

Obzirom da smo u današnje vreme zatrpani sadržajem na internetu koji je često jako loše struktuiran postoji objektivna potreba za aplikacijama ovog tipa koje će omogućiti struktuiranje i deljenje sadržaja među korisnicima i time smanjiti vreme pronalaženja kvalitetnog sadržaja iz željene oblasti.

7. Literatura

- [1.] Matthias Noback: "A Year With Symfony", Leanpub, 2014
- [2.] <http://www.symfony.com>
- [3.] <http://twig.sensiolabs.org/>
- [4.] http://symfony.com/doc/current/best_practices/index.html
- [5.] <http://getbootstrap.com/>
- [6.] <http://www.chartjs.org/docs/>
- [7.] <http://alexgorbatchev.com/SyntaxHighlighter/>
- [8.] <http://materializecss.com/>
- [9.] <http://richardmiller.co.uk/2012/10/31/symfony2-trimming-fat-from-controllers/>
- [10.] <https://api.stackexchange.com/docs>