Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Pangerc

Razvoj in implementacija orodja za urejanje spletne vsebine

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

izr. prof. dr. Narvika Bovcon
MENTOR

as. Blaž Meden
SOMENTOR

Ljubljana, 2019

Tematika naloge:

V diplomski nalogi preglejte obstoječe urejevalnike spletne vsebine. Razvijte svoj sistem in naročniku ponudite izbrane funkcionalnosti, pri čemer poskrbite za čim lažje urejanje. Opišite uporabljene tehnologije in delovanje sistema.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani izjavljam, da sem avtor dela, da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali drugem visokošolskem zavodu, razen v primerih kjer so navedeni viri.
S svojim podpisom zagotavljam, da:

- sem delo izdelal samostojno pod mentorstvom izr. prof. dr. Narvika Bovcon in somentorstvom as. Blaž Meden,
- so elektronska oblika dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko in
- soglašam z javno objavo elektronske oblike dela v zbirki "Dela FRI".

— Nejc Pangerc, Ljubljana, februar 2019.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Pangerc

Razvoj in implementacija orodja za urejanje spletne vsebine

POVZETEK

Diplomsko delo opisuje razvoj vizualnega urejevalnika spletne vsebine. Prilagojen je potrebam stranke in brez omejitev sistema za nadaljnji razvoj. Opisuje sisteme, ki so trenutno v uporabi, opisuje tudi tehnologije, ki se uporabljajo pri razvoju sistema. Sistem bazira na C# programskem jeziku v MVC arhitekturi. Za pomoč pri prikazovanju in razvoju nam služijo še drugi programski jeziki, kot so HTML, CSS, JavaScript, jQuery in SQL. Sistem deluje na Windows operacijskih sistemih na IIS strežniku.

Ključne besede: sistem za urejanje spletne vsebine, spletna stran, razvoj programske opreme, C#

University of Ljubljana
Faculty of Computer and Information Science

Nejc Pangerc

Development and implementation of content management system

ABSTRACT

The thesis describes the development of the visual web content management system. It is tailored to the needs of the customer and without limitations of the system for further development. It describes the systems currently in use as well as the technologies used in the development of the system. The system is based on the C# programming language with the MVC architecture. Other programming languages, such as HTML, CSS, JavaScript, jQuery and SQL, help us show and develop the system. The system works on Windows operating systems on the IIS server.

Key words: web content management system, website, software development, C#

KAZALO

Slike

Tabele

Algoritmi

1 Uvod

1.1 Problem

Živimo v informacijski dobi, kar pomeni, da nam tehnologija pomaga pri vsakodnevnih opravilih. To doprinese k hitrejšemu opravljanju zadanih nalog. Z njo si pomagamo skoraj na vsakem koraku saj si življenja brez uporabe vseh pripomočkov ne predstavljamo več. Tudi vsako podjetje skuša prodreti na trg preko teh kanalov, zato pridemo do problema, da si podjetja želijo oglaševanja na svetovnem spletu.

Za mnoga podjetja izdelava spletne strani predstavlja velik izziv, zato to prepuščajo strokovnjakom, ki se spoznajo na izdelavo spletnih strani. Podjetje si želi imeti spletno strani, ki bo ažurna in urejena, zato pri tem naletimo na problem.

Problema se lahko lotimo tako, da zopet zaposlimo programersko skupino. Vendar je to lahko zamudno, ker čakamo na njihov odziv. Tak način je tudi drag, ker moramo za vsako spremembo poklicati in plačati programerja. Tako nam preostane uporaba sistema za urejanje spletne vsebine (angl. CMS Content Managment Sistem).

Na spletu obstaja že mnogo obstoječih sistemov, kot so Wordpress, Joomla, Durpal itd. Vendar pri vseh naletimo na problem, ker uporabnik ne opazi spremembe vizualno,

ampak mora ponovno osveževati spletno stran Na problem naletimo tudi, ko želimo imeti neko nestandardno rešitev, ki pa ni podprta v omenjenih sistemih.

1.2 Rešitev

Cilj diplomske naloge je razviti sistem za urejanje spletne vsebine, s katerim bi zagotovili lažje in hitrejše urejanje spletnega mesta. S tem sistemom bi se tudi omogočile dodatne funkcionalnosti, ki pri bolj znanih sistemih niso podprte. Cilj je, da lahko vsak tudi nevešč uporabnik ureja, in spreminja spletno stran, kar je prednost za podjetja, ki nimajo potrebnega znanja za njihovo urejanje.

Sistem bo omogočal urejanje vsebine s preprostim klikom na izbrani del strani, besedilo ali sliko in nam omogočil urejanje. Po končanem urejanju ne bo potrebne ponovne osvežitve strani, ampak le preprost klik na gumb in že bomo videli stran, kot jo vidijo obiskovalci. To pripomore k lažjemu upravljanju in predstavlja naprednejšo interakcijo z uporabnikom.

Pri razvoju sistema za upravljanje spletne vsebine nam bodo v pomoč že obstoječi sistemi in njihove prednosti ter omejitve, ki jih bomo poskušali odpraviti v lastnem sistemu za upravljanje spletne vsebine.

2 Opis problema

2.1 Kaj je Sistem za urejanje spletne vsebine?

CMS (Content Management Sistem) ali po slovensko sistem za urejanje spletne vsebine je programska oprema, ki omogoča uporabnikom z malo znanja o programskih jezikih ali označevalnih jezikih lahko in enostavno urejanje, spreminjanje, brisanje vsebine ter vzdrževanje spletne strani.

Taki sistemi so se začeli razvijati proti koncu leta 1990. CMS so v veliki meri uporabljeni pri spletnih straneh, ki vsebujejo bloge, novice ali pa so namenjene nakupovanju. Veliko podjetij uporablja CMS. Prednost sistemov je v tem, da od uporabnika ne zahtevajo znanja programiranja [1].

2.2 Glavne funkcionalnosti sistemov

Osnovna funkcionalnost sistemov je shranjevanje in urejanje datotek. Obstajajo različno kompleksne različice sistemov, odvisno od potreb podjetja oz. uporabnika.

Poznamo dva osnovna koncepta sistemov za urejanje:

- Content managment aplication (CMA)
- Content display aplication (CDA)

WCMS (Web Content Managment Sistem) se uporablja za upravljanje dinamičnih spletnih, vsebin vključno s HTML dokumenti, slikami in drugimi medijskimi viri.

2.2.1 Tak sistem po navadi omogoča

- izbiro predlog spletne vsebine in lahko spreminjanje le-te.
- da imajo dostop do urejanja strani imajo samo določeni uporabniki, navadni obiskovalci strani ne morejo dostopati do CMS.
- lahko urejanje vsebine.
- lahko dodajanje dodatnih modulov k spletni strani.
- lažjo nadgraditev spletne strani.
- različne nivoje pravic, ker ni potrebno, da imajo vsi uporabniki iste pravice urejanja.
- urejanje dokumentov.
- lahko spreminjanje jezika.

2.2.2 Poznamo tri glavne tipe takih sistemov

- Offline urejanje strani: stran se ureja na uporabnikovi strani šele po prenosu na strežnik je stran vidna tudi uporabnikom spleta.
- Online urejanje strani: spremembe so vidne takoj po osvežitvi strani. Primerno za bloge, slike, besedila itd.
- Hibrid: je mešanica prvih dveh tipov.

2.2.3 Prednosti

- Nizka cena – nekateri sistemi so zastonj, najbolj znani so Drupal, Wordpress in Joomla. Zaradi uporabe takega sistema nam ni treba zaposliti razvijalca spletne strani.
- Lahko urejanje in spreminjanje vsebine.
- Lahka uporaba neusposobljenim kadrom.
- Hierarhija uporabe strani.
- Dobri za optimizacijo.

2.2.4 Slabosti

- Cena razvoja.
- Cena vzdrževanja.
- Pri velikem obisku strani pride do preobremenitve sistema.
- Mešanje orodij.
- Varnost.
- Problem pri nestandardnih rešitvah.

2.3 Opis nekaj najbolj znanih sistemov za urejanje

V uporabi je že veliko sistemov, vsi imajo svoje prednosti in slabosti. Največji problem pa nastane pri nestandardnih reštivah, ki si jih zaželi stranka. V nadaljevanju bomo pregledali nekaj najbolj razširjenih obstoječih rešitev in pregledali njihove prednosti in slabosti.

2.3.1 Wordpress

Wordpress je zastonj in odprtokodni sistem za urejanje spletne vsebine, ki temelji na PHP programskem jeziku, za podatkovno bazo pa uporablja MySQL. Wordpress je najbolj priljubljen sistem za upravljanje z blogi in je uporabljen na več kot 74 milijonov spletnih straneh [2]. Izdan je bil leta 2003. Wordpress temelji na predlogah (template) in uporablja ≫template processor≪.

Lastnosti Wordpressa:

- Wordpress teme: Uporabniki si jih lahko poljubno izberejo in spreminjajo. Lahko izbirajo med zastonjskimi in plačljivimi temami ali pa jih ustvarijo sami. Teme lahko tudi poljubno uredijo.
- Wordpress vtičniki (plugins): Arhitektura sistema omogoča uporabnikom dodajanje različnih vtičnikov za razširjanje lastnosti. Problem nastane pri posodobitvi sistema.
- Wordpress podpira tudi mobilne naprave, tako omogoča mobilnemu obiskovalcu strani prilagojeno stran.

Ranljivost sistema

Pomanjkljivosti pri sistemu odpravljajo vendar problem nastane, ker uporabniki ne posodobijo svojih strani in tako ostanejo ranljivi na napade. Zato so pri verziji 3.7 predstavili avtomatsko nadgrajevanje sistema.

Samostojno inštalacijo sistema lahko zavarujemo z različnimi varnostnimi vtičniki. Najbolj se zavarujemo, če redno posodabljamo vse elemente sistema in ne uporabljamo tem in vtičnikov iz neznanih virov.

Razvijalci lahko uporabijo orodje, ki analizira možne pomanjkljivosti in nas na to opozori, vendar orodje ne odkrije vseh pomanjkljivosti. Zato svetujejo pregled drugega razvijalca [3].

Slabosti sistema

- Večje spreminjanje sistema zahteva znanje PHP, ki pa je zamudno in drago.
- Grafično spreminjanje izgleda strani zahteva znanje HTML in CSS.
- Da dosežemo funkcionalnost drugih sistemov, potrebujemo veliko vtičnikov. Problem nastane pri neujemanju vtičnikov in pri izbiri pravega za naš problem.
- Ranljivost PHP sistemov – sistemi PHP so manj zavarovani kot drugi podobni sistemi.
- Težje urejanje tabel.
- SQL poizvedbe zahtevajo kompleksne rešitve.

Zaključek

Wordpress je idealen za začetnike in nezahtevne uporabnike. Je lahek za uporabo, vendar zahtevnejši uporabniki naletijo na omejitve z vtičniki, kar pomeni daljši in dražji razvoj.

2.3.2 Drupal

Drupal je prav tako kot Wordpress zastonj in je odprtokodni sitem za urejanje spletne vsebine. Napisan je v PHP in distribuiran pod GNU licenco. Uporablja se za bloge in za strani raznih organizacij. Je enostaven za uporabo, saj ga večinoma namestimo kar iz brskalnika [4].

Funkcinalnosti ki jih vsebuje

- Moduli – skripte ki izboljšajo stran.
- Teme – različni izgledi strani.
- Bloki – to so vidni elementi spletne strani.
- Čisti URL – brez podatkov v URL naslovu.
- Urejanje člankov.
- Urejanje pravic različnim uporabnikom.

Varnost

Drupal nam sporoči za vsako varnostno napako, ko je popravljena. Administratorji so o posodobitvi obveščeni avtomatsko, tako da lahko hitro odpravijo napako na svojem sistemu.

Slabosti sistema

- Uporabnost – uporabniški vmesnik je nepregleden.
- Učljivost – je težko učljiv.
- Nekompatibilnost verzij – starejše teme in moduli ne delujejo v novejših verzijah, zato morajo razvijalci spreminjati kodo, kar pa je lahko zamudno.
- Zmogljivost – Wordpress je mnogo hitrejši kot Drupal.
- Itegrabilnost s strežniki – zaradi poizvedovanja lahko pri večjem obisku preveč obremenimo strežnik.
- Iskalnik – ima slab iskalnik po spletni strani.
- Veliko nedokončanih in zastarelih modulov.

Zaključek

Drupal je primeren, ker omogoča izdelovanje najbolj naprednih strani, ampak samo za razvijalce z veliko znanja na tem področju.

2.3.3 Joomla

Tudi Joomla je zastonj in je odprtokodna rešitev sistema. Prav tako kot Wordpress je sistem napisan v PHP programskem jeziku. Za podatkovno bazo uporablja MySQL, MS SQL ali PostgreSQL, odvisno od potreb stranke [5].

Prednosti

- Lahko učljiv.
- Lahka namestitev sistema.
- Vtičniki – obstaja veliko vtičnikov za razširitev sistema.

Slabosti sistema

- Napredni vtičniki so lahko dragi.
- Vtičniki ponavadi potrebujejo ponovno programiranje.
- Upravljanje pravic je neustrezno.

Zaključek

Joomla je primeren za izdelavo spletnih trgovin, vendar potrebuje razvoj vtičnikov, da zadostimo potrebam stranke.

3 Opis tehnologij

3.1 SQL

SQL (Structured Query Language) ali strukturirani poizvedovalni jezik je posebni jezik, ki je bil razvit za upravljanje s podatki in bazami podatkov. Je najbolj razširjen poizvedovalni jezik, ki je določen s standardom ANSI/ISO. Razvijati so ga začeli leta 1986 in do danes se je razvilo že mnogo različic [6].

Najbolj znane poizvedbe se vršijo s stavki, kot so

- `SELECT` – stavek nam vrne podatke iz ene ali več tabel. Stavek ne vpliva na podatke v bazi.
- `INSERT` – s tem stavkom dodajamo vrstice v tabelo
- `UPDATE` – s tem stavkom spreminjamo podatke v tabeli
- `DELETE` – s tem stavkom izbrišemo vrstico iz tabele
- Po izvedbi katerega koli zgoraj naštetega stavka moramo zahtevo potrditi (`COMMIT`) ali razveljaviti (`ROLLBACK`).

3.2 SQL Server Management Studio

SQL Server Managment Studio je program, ki se uporablja za upravljanje, nastavljanje in administracija komponent Microsoft SQL Serverja. Orodje uporablja grafični in skriptni način [7].

S programom se preko naslova SQL strežnika ter uporabniškega imena in gesla povežemo na SQL strežnik in tako dobimo dostop do podatkovnih baz. Vsaka podatkovna baza je skupek med seboj povezanih tabel, v katerih so podatki. V programu lahko pišemo SQL stavke, preko katerih dostopamo do podatkov ali jih spreminjamo. Program nam tudi omogoča izvoz podatkov in arhiviranje podatkovne baze, če potrebujemo prenos na produkcijski strežnik ali za varnostno kopijo.

3.3 Linq to SQL

LINQ (Language Integrated Query) je komponenta Microsoft .NET Frameworka, ki nam omogoča lažje dostopanje in urejanje podatkov v podatkovni bazi. LINQ to SQL prevede LINQ poizvedbe v SQL poizvedbe, ki so nato poslane na SQL server. LINQ nam omogoča programiranje v izbranem jeziku, tudi če ne obvladamo SQL stavkov [8].

3.4 JavaScript

JavaScript ali JS je objektni skriptni jezik, ki ga je razvil Netscape okoli leta 1995. Razvit je bil za ustvarjanje interaktivnih spletnih strani. Kljub imenu je bil jezik razvit neodvisno od Jave, ampak z Javo si delita nekatere strukture in lastnosti [11].

JS lahko uporabljamo skupaj s HTML-kodo kar pomaga pri ustvarjanju dinamičnih spletnih strani. JS je odprtokodni jezik, kar pomeni da ga lahko uporablja kdorkoli.

JS je bolj kot Javi podoben programskemu jeziku C. Programski jezik je podprt v večini današnjih brskalnikov in operacijskih sistemih [12].
Najpogostejša uporaba JavaScripta:

- nalaganje novih komponent spletne strani s pomočjo AJAX-a brez ponovnega nalaganja strani;
- interaktivnost – razne igre, video in zvok;
- iskalniki;

- pošiljanje podatkov;
- animacija delov strani.

3.5 jQuery

jQuery je JavaScript knjižnica, razvita za lažje pisanje skript. Je hiter, majhen in funkcionalen. S prihodom jQuerya se je spremenilo pisanje JavaScript skript. Je najbolj popularna odprtokodna knjižnica, ki je v uporabi. Sintaksa omogoča lažje sprehajanje po DOM drevesu, ustvarjanje animacij, upravljanje dogodkov in razvoj AJAX aplikacij [9] [10].

3.6 AJAX

Osnovni namen razvoja AJAX tehnologije je pošiljanje podatkov med strežnikom in klientom na uporabnikovi strani brez ponovnega nalaganja spletne strani. Vse to se dogaja v ozadju, ne da bi uporabnik vedel za to [13].

AJAX lahko uporabljamo skupaj s HTML in CSS. S tem konceptom smo dosegli hitrejše in bolj tekoče spletne strani. Zmanjšal se je tudi promet med strežnikom in brskalnikom, kar pomeni manj obremenjene strežnike. Podatke pošiljamo s pomočjo XML, JSON ali HTML objktov.

Na problem pa naletimo pri navigaciji po strani, ker se podatki osvežujejo samo na trenutni spletni strani.

3.7 HTML

HTML je označevalni jezik za opisovanje spletnih elementov oz. spletnih strani. Je osnova za spletno stran. Sestavljen je iz značk, ki označujejo posamezen element strani.

Značka je sestavljena iz začetne in končne značke, med nju pa lahko vstavimo poljubno vsebino, odvisno od značke. Značkam lahko dodajamo tudi atribute in stile. Za uporabo ne potrebujemo nobenega razvojnega okolja, ker lahko dokument napišemo kar v beležnici [14] [12].

3.8 CSS

CSS ali kaskadne stilske podloge so podloge (sheets), ki so napisane v preprostem jeziku in skrbijo za stilski prikaz spletne strani. Uporablja se za stilizacijo HTML elementov. Stili so določena pravila, kako prikazati element [12].

S prihodom CSS so se stili iz HTML prestavili v CSS datoteke. Poenostavilo se je tudi večkratno ponavljanje stilov dedovanje stilskega prikaza.

V CSS datotekah lahko določamo barve, velikost pisave, obrobe, itd. Prednost pa je predvsem v tem, da so se stili ločili iz HTML dokumentov, kar pripomore k večji preglednosti dokumentov in lažje popravljanje oz. spreminjanje stilov [15].

3.9 IIS

IIS je razširljiv spletni strežnik, ki ga je razvil Microsoft. Za APACHE strežnikom je to drugi najbolj priljubljen strežnik. Uporablja se lahko za spletni ali FTP strežnik.

Vsebuje orodja za razvijanje in administracijo strani. Prav tako pa nudi podporo spletnim aplikacijam, ki se povezujejo s podatkovno bazo.

IIS je močno povezan z operacijskim sistemom Windows NT in 2000 Servers, rezultat tega je hitrejše delovanje spletnih strani [16].

3.10 ASP.NET

ASP je tehnologija, ki omogoča poganjanje dinamičnih spletnih strani in se izvaja na strežniku. ASP.NET je nova generacija ASP, glavna razlika je, da so spletne strani prevedene, kar jih naredi hitrejše.

Navadno so strani napisane v Visual Basic ali C#. Ob zahtevi za ASP.NET datoteko se ta prebere, prevede in izvrši skripte v datoteki, nato pa vrne rezultate brskalniku v čisti HTML obliki [8].

3.11 C# (C Sharp)

C# je programski jezik ki ga je razvil Microsoft in je objektno orientiran. C# naj bi bil naslednik programskega jezika C++, kljub temu da vsebuje veliko značilnosti Jave. Razvit je bil posebej za delo z .NET platformo. Platformo srečamo v .NET orodjih, aplikacijah tako za osebni računalnik kot mobilne telefone.

Namen Microsofta je bilo ustvariti jezik, ki bi bil lahek za uporabo, moderen, namenjen za splošno uporabo in objektno orientiran jezik. Programska robustnost, zanesljivost in programerska produktivnost so bili cilji razvijalcev. C# aplikacije so ekonomične s pomnilnikom in procesorskim zahtevami [8].
Glavne prednosti jezika C#:

- Prenosljivost - C# prevajalnik bi lahko v teoriji generiral strojno kodo, podobno kot tradicionalni prevajalniki za C++ ali Fortran.
- Hitrost programiranja - C# je varnejši kot C++, saj ne dopušča spreminjanja tipa spremenljivke. C# nima globalnih spremenljivk, vse spremenljivke morajo biti v nekem razredu. Vse metode morajo biti deklarirane v razredih.
- Namespaces - z njimi ločimo dele kode.

3.12 Razvojno okolje Visual Studio

Visual Studio je razvojno okolje (IDE), ki ga je izdalo ameriško podjetje Microsoft. Njegov namen je bil predvsem razvoj programov za Windows sisteme, razvoj aplikacij, spletnih strani ter ostalih aplikacij in storitev, ki bazirajo na ASP.NET frameworku.

Podpira programiranje tako strojne in upravljalne kot tudi interpretirane kode. Ima tudi možnost razširitve preko vtičnikov, eden najbolj poznanih je Xamarin, ki se uporablja za programiranje mobilnih aplikacij.

Program podpira več kot 36 programskih jezikov, med katerimi so najbolj poznani: C, C++, C#, Visual Basic, Fortran, JavaScript, HTML, CSS, Python in drugi.

Osnovna verzija programa se imenuje Visual Studio Comunity edition in je brezplačna za uporabo.

Ima tudi možnost povezave tako na TeamFundationServerje kot na Git strežnike, kar olajša delo podjetjem in programerskim skupinam in nudi nadzor nad verzijami kode.

Urejevalnik kode podpira InteliSense, kar nam omogoča lažje programiranje in lažje razhroščevanje [17].

3.13 MVC arhitektura

Koncept MVC ali Model-View-Controller je arhitekturna osnovna, ki se ponavadi uporablja pri razvoju uporabniških vmesnikov in aplikacij.

Aplikacija je razdeljena na 3 povezane dele. Deli so ločeni zaradi načina pridobivanja in prikaza informacij.
MVC arhitektura je sestavljena iz:

- modelov (Model) - zbirajo podatke, ki se potem preko pogledov prikažejo uporabniku;
- pogledov (View) - definirajo, kako se bodo podatki prikazovali uporabniku. Prikazujejo se podatki iz modelov;
- kontrolerjev (Controller) - vsebujejo logiko, ki posodablja in prikazuje podatke glede na interakcijo uporabnika.

Arhitektura se ponavadi uporablja pri grafičnih uporabniških vmesnikih, najbolj popularna pa je arhitektura postala pri razvoju spletnih aplikacij.

Najbolj popularni jeziki, ki uporabljajo MVC arhitekturo so Java, C#, Python, Ruby, PHP.

Cilji razvoja MVC so sočasen razvoj, uporaba iste kode večkrat. [18]
Prednosti uporabe MVC:

- sočasen razvoj – več programerjev lahko na enkrat programira sistem;
- nizka povezanost – arhitektura MVC je naravnana tako, da je povezanost med modeli, pogledi in kontrolerji nizka;
- enostavno spreminjanje – ločevanje nam omogoča lažje programiranje;
- visoka kohezija – MVC omogoča logično združevanje na kontrolerju;
- več pogledov na model – vsak model ima lahko več pogledov;
- ni ponavljanja kode – ista koda se lahko uporabi večkrat;

Slabosti uporabe MVC:

- slaba navigacija po kodi;
- programerji morajo vzdrževati več elementov naenkrat;
- sprotno učenje – razvijalci morajo biti usposobljeni za več tehnologij in jih nadgrajevati.

4 Načini rešitve

V nadaljevanju poglavja bom opisal način razvoja lastnega urejevalnika spletne vsebine ali CMS, pri katerem sem uporabil v prejšnjem poglavju naštete tehnologije.
Postopek izdelave CMS:

- postavitev baze podatkov za shranjevanje vsebine;
- izbira predloge spletne strani;
- vdelava predloge v ASP.NET MVC projekt;
- izdelava urejevalnika (CMS);
- urejanje spletne strani preko CMS.

4.1 Podatkovna baza

Podatkovna baza hrani podatke, ki omogočajo delovanje spletne strani. Iz nje se pridobivajo podatki o vsebini, to so besedila in slike. V bazi imamo tudi zapise o uporabnikih, ki lahko dostopajo do urejanja. Podatkovno bazo si lahko predstavljamo kot skupek

logično povezanih podatkov s katerimi zadostimo potrebam aplikacije. Podatkovno bazo sestavljajo tabele, vsaka tabela pa je sestavljena iz več atributov.

Glavne tabele podatkovne baze so tabele, ki hranijo podatke o vsebini, in tabela z uporabniškimi podatki za dostop do sistema.

4.1.1 Shema entitete Page

V tabeli `Page` se nahajajo vse podstrani, ki se prikazujejo. Na to tabelo je tudi vezana vsebina. Entiteto tabele `Page` dobimo preko URL parametrov, in sicer preko kotrolerja, akcije in morebitnih parametrov. S pomočjo te entitete lahko preko njenega `Id` atributa pridobimo nadaljnjo vsebino spletne strani.

Zaporedna številka	Atribut	Tip atributa	Opis atributa
1	`Id`	celo število	primarni ključ tabele
2	`Controler`	besedilo	ime kontrolerja
3	`Action`	besedilo	ime akcije
4	`QueryId`	besedilo	parameter id.
5	`QueryParams`	besedilo	dodatni parametri strani
6	`ParentId`	celo število	`Id` nad strani*
7	`Title`	besedilo	ime strani
8	`Type`	celo število	tip strani

Opombe: * V primeru NULL je stran primarna.

Tabela 4.1: Shema entitete `Page`

4.1.2 Shema entitete PageSection

V tabeli `PageSection` se nahaja vsa vsebina, ki se prikazuje na spletni strani. V tabeli preko polja `PageId` izvemo, na kateri strani se prikazuje izbrana vsebina. Preko entitete `Page` in polja `Id` dobimo lahko vse segmente za izbrano stran. To naredimo tako, da iz baze izberemo vse entitete `PageSection`, kjer je njihovo polje `PageId` enak `Id` polju `Page` entitete. Za prikaz določene vsebine nam služi atribut `Title` in `ItemTypeId`, ki nam pove, za kateri tip vsebine gre, preko njega izvemo, v kateri tabeli moramo iskati vsebino. V tabeli imamo tudi zapisano, kdaj je bila vsebina narejena, kdaj je bila nazadnje spremenjena in kateri uporabnik jo je spremenil.

Zaporedna številka	Atribut	Tip atributa	Opis atributa
1	`Id`	celo število	primarni ključ tabele
2	`Title`	besedilo	ime vsebine
3	`PageId`	celo število	`Id` strani, na kateri se prikazuje
4	`ItemTypeId`	celo število	tip vsebine*
5	`ItemId`	celo število	`Id` vsebine
6	`ModifiedUserId`	celo število	`Id` uporabnika, ki je spremenil vsebino
7	`UpdatedDate`	datum	datum spremembe
8	`CreatedDate`	datum	datum kreiranja

Opombe: * 1 - besedilo, 2 - obogateno besedilo, 3 - slika.

Tabela 4.2: Shema entitete `PageSection`

4.1.3 Shema entitete PageSectionText

V tabeli PageSectionText se nahaja vsebina tipa 1 - navadno besedilo. Vsebino iz tabele pridobimo preko `Id` polja tabele `PageSection`, ki je enak polju `SectionId` ter preko polja `LanguageId`, ki ustreza jeziku prikazane spletne strani. Vsebina se nahaja v polju `Value`, vrnemo jo kot navadno besedilo.

Zaporedna številka	Atribut	Tip atributa	Opis atributa
1	`Id`	celo število	primarni ključ tabele
2	`Value`	besedilo	besedilo
3	`LanguageId`	celo število	`Id` jezika
4	`SectionId`	celo število	`Id` iz `PageSection` tabele

Tabela 4.3: Shema entitete `PageSectionText`

4.1.4 Shema entitete PageSectionRichText

V tabeli `PageSectionRichText` se nahaja vsebina tipa 2 - obogateno besedilo. Vsebino iz tabele prav tako pridobimo preko `Id` polja tabele `PageSection`, ki je enaka polju `SectionId`, in preko polja `LanguageId`, ki ustreza jeziku prikazane spletne strani. Vse-

bina se nahaja v polju `Value` in jo vrnemo kot HTML besedilo.

Zaporedna številka	Atribut	Tip atributa	Opis atributa
1	`Id`	celo število	primarni ključ tabele
2	`Value`	besedilo	besedilo
3	`LanguageId`	celo število	`Id` jezika
4	`SectionId`	celo število	`Id` iz PageSection tabele

Tabela 4.4: Shema entitete `PageSectionRichText`

4.1.5 Shema entitete PageSectionPicture

V tabeli `PageSectionPicture` se nahaja vsebina tipa 3 - slike. Vsebino iz tabele prav tako pridobimo preko `Id` polja tabele `PageSection`, ki je enaka polju `SectionId`, in preko polja `LanguageId`, ki ustreza jeziku spletne strani. Vsebino oz. sliko vrnemo kot pot do slike na disku, pot sestavimo v aplikaciji skupaj s pomočjo polja `Name`, ki vsebuje ime slike. V tabeli se nahaja še datum vstavljanja slike in alternativno ime slike, ki se prikaže v primeru, da se slika ne naloži. Polje `GuId` nam služi kot generirano ime slike, ki je unikatno, da ne pride do prekrivanj.

Zaporedna številka	Atribut	Tip atributa	Opis atributa
1	`Id`	celo število	primarni ključ tabele
2	`AlternativeText`	besedilo	napis v primeru napake slike
3	`LanguageId`	celo število	`Id` jezika
4	`SharedGuId`	besedilo	generiran GuId slike
5	`GuId`	besedilo	generiran GuId slike
6	`SizeId`	celo število	velikost slike
7	`InsertedDate`	datum	datum kreiranja
8	`SectionId`	celo število	`Id` iz `PageSection` tabele
9	`Name`	besedilo	ime slike

Tabela 4.5: Shema entitete `PageSectionPicture`

4.1.6 Shema entitete User

V tabeli `User` se nahajajo podatki o uporabnikih, ki lahko uporabljajo aplikacijo. Uporabnika pridobimo preko vpisane e-pošte in gesla na strani za vpis. Od tu naprej se za identifikacijo uporabnika uporablja polje `Id`, ker je unikatno, tako lahko natančno določimo uporabnika. V tabeli so zapisani še ime, priimek, datum vpisa v sistem in status, ki nam služi za preverjanje, če je uporabnik še aktiven.

Zaporedna številka	Atribut	Tip atributa	Opis atributa
1	`Id`	celo število	primarni ključ tabele
2	`Email`	besedilo	e-pošta uporabnika
3	`Password`	besedilo	kriptirano geslo
4	`FirstName`	besedilo	ime uporabnika
5	`LastName`	besedilo	priimek uporabnika
6	`Inserted`	datum	datum kreiranja
7	`Status`	celo število	status uporabniškega računa

Tabela 4.6: Shema entitete `User`

4.2 Spletna stran

Dobro narejena spletna stran je lahko konkurenčna prednost. Lična ter uporabnikom prijazna spletna stran daje obiskovalcem strani več informacij in posledično lahko to pomeni več uspeha na področju, ki ga spletna stran predstavlja. Vendar je potrebno paziti, da uporabnika ne zasipamo z nepotrebnimi informacijami. Današnja družba je namreč naravnana tako, da bi rada čim prej prišla do potrebnih informacij.

Lastnosti dobrih spletnih strani:

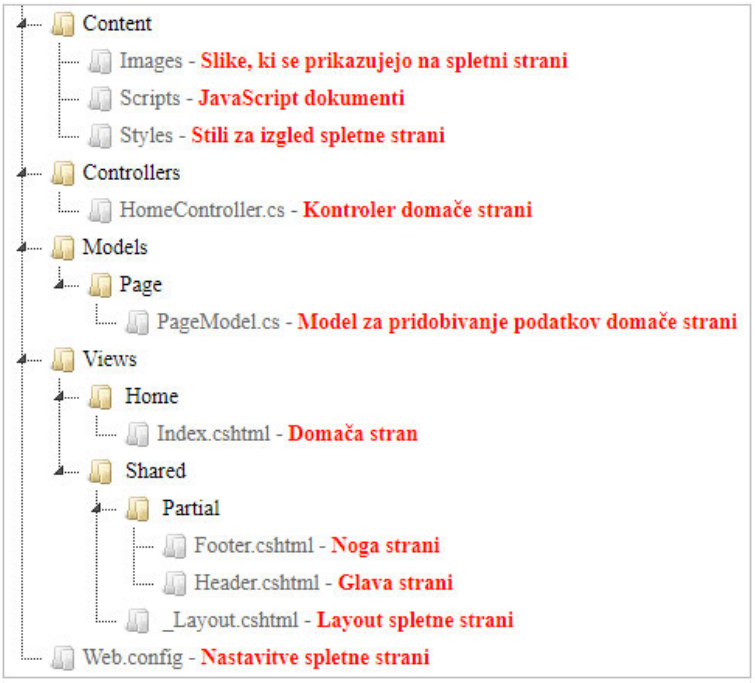
- Najprej vsebina, šele nato oblika.
- Aktualna vsebina, kar pomeni redno objavljanje novic, projektov itd.
- Hitrost nalaganja vsebine.
- Rangiranje na iskalnikih, kot so google ...

4.2.1 Izbira predloge spletne strani

Pred začetkom postavitve spletne strani si izberemo, kako naj bi naša spletna stran izgledala, pregledati moramo tudi želje in zahteve za spletno stran, da lahko najdemo pravilno predlogo, ki bo vsebovala vse gradnike, ki so nam potrebni. Dobro je razmišljati še o prihodnosti spletne strani in če je predloga optimizirana za mobilne naprave, da se v prihodnjem razvoju izognemo dodatnemu delu. Po izboru predloge si gradnike preuredimo tako, da zadostijo zahtevam.

4.2.2 Vdelava predloge v ASP.NET MVC projekt

Izbrano predlogo vdelamo v ASP.NET MVC projekt, kar pomeni da si postavimo MVC projekt, v katerem imamo kontrolerje kot imena podstrani in modele, s katerimi pridobivamo podatke iz podatkovne baze. V "poglede- Views vstavimo našo predlogo, ki smo si jo pred tem pripravili. Dodamo JavaScript dokumente in CSS stilske datoteke. S tem pridobimo izgled spletne strani, kot smo si ga pripravili v predlogi.



Slika 4.1: Struktura projekta

4.2.3 Pridobivanje podatkov iz baze podatkov

Za pridobivanje vsebine iz baze podatkov uporabljamo 3 različne metode, in sicer:

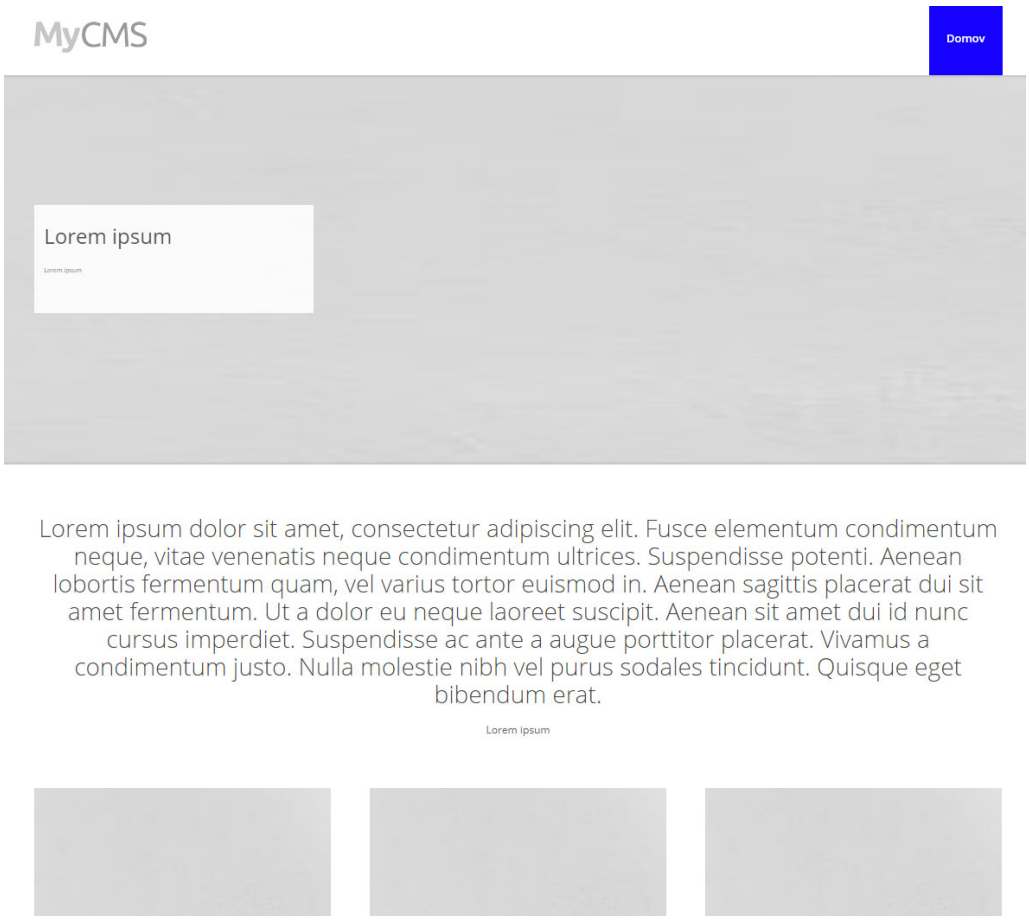
- `GetPageSectionText("ime vsebine")` - to metodo uporabljamo za pridobivanje navadnega besedila;
- `GetPageSectionRichText("ime vsebine")` - to metodo uporabljamo za pridobivanje daljšega besedila skupaj z njegovimi stili;
- `GetPageSectionPicture("ime vsebine", "širina", "višina")` - to metodo uporabljamo za pridobitev slike, v metodo dodamo še širino in višino slike.

Metoda za pridobivanje vsebine nam vrne HTML element, v katerem se nahaja vsebina glede na tip metode, ki smo jo uporabili, in ime vsebine. Če izbrana vsebina ne obstaja v bazi, jo sistem doda avtomatsko, in sicer z vzorčno vsebino.

Vso vsebino, ki jo želimo urejati zamenjamo z izbrano metodo, tako se spletna stran postavi z vzorčno vsebino, ki jo bomo v nadaljevanju urejali.

```
public MvcHtmlString GetPageSectionText(string title)
{
    //Nalaganje vsebine iz baze
    PageSection section = Db.PageSections.FirstOrDefault(x => x.Title == title && x.
        PageId == this.Id && x.ItemTypeId == 1);
    //V primeru da vsebina se ne obstaja vnesemo novo vsebino z vzorcno vsebino.
    if (section == null)
    {
        section = NewPageSection(section, title, 1);
        PageSectionText textSection = new PageSectionText();
        textSection.SectionId = section.Id;
        textSection.Value = "Lorem ipsum";
        textSection.LanguageId = 1;

        Db.PageSectionTexts.InsertOnSubmit(textSection);
        Db.SubmitChanges();

        section.ItemId = textSection.Id;

        Db.SubmitChanges();
    }

    //Nalozimo vsebino iz podatkovne baze.
    PageSectionText sectionText = Db.PageSectionTexts.FirstOrDefault(x => x.Id ==
        section.ItemId);

    return new MvcHtmlString(GetParSection(section.Title, sectionText.Value, 1));
}

public string GetParSection(string title, string value, int itemTypeId)
{
    //Vrni sestavljen HTML element, ki vsebuje gumb za urejanje in vsebino, ki smo jo
        pridobili iz baze.
    return string.Format(@"<div class=""section-container"">"
        + @"<par class=""section-paragraph"" data-section-title=""{0}"" data-page-id="
            "{1}"" data-section-type=""{3}"">"
        + @"<div class=""editButton""><div class=""text""><i class=""icon-edit""></i>
            Uredi</div></div>"
        + "{2}"
        + @"</par>"
        + "</div>", title, this.Id, value, itemTypeId);
}
```

Algoritem 1: Metoda za pridobivanje besedila



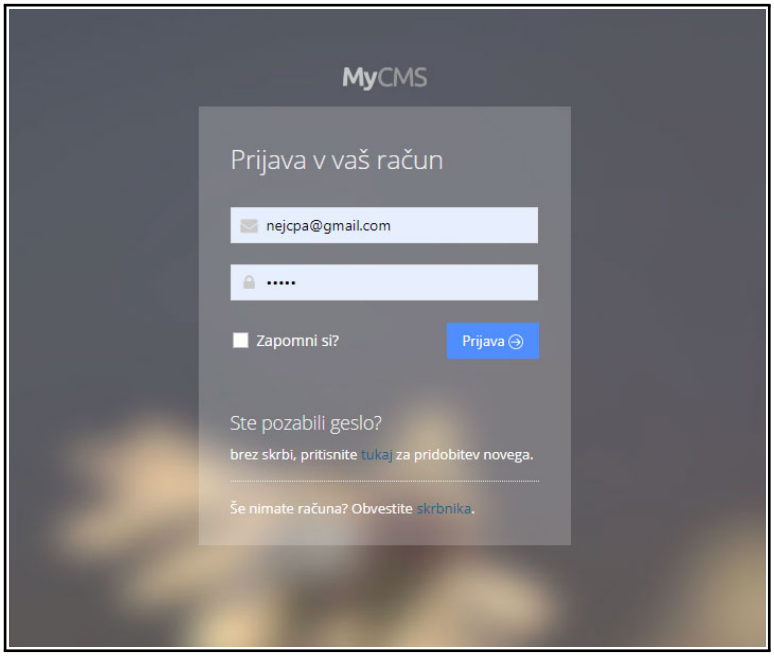
Slika 4.2: Izgled vzorčne spletne strani

4.3 Izdelava urejevalnika spletne vsebine

Uporabniški vmesnik spletne aplikacije mora biti preprost za uporabo, tako da uporabniki nimajo težav z njegovo uporabo. Upravljalec spletne strani se s pomočjo uporabniškega imena in gesla vpiše v sistem. Uporabniku se po vpisu v sistem prikaže spletna stran takšna, kot jo vidijo navadni uporabniki, s to razliko, da se v aplikaciji prikažejo gumbi za urejanje vsebine. Vsebino urejamo s klikom na gumb "uredi", ki se pojavi ob prehodu z miško skozi vsebino. Po kliku se pojavi modalno okno, v katerem glede na tip urejamo vsebino. Po zaključku urejanja kliknemo na gumb šhrani"in sistem shrani vsebino v bazo, nam pa prikaže novo vsebino, ki se tudi takoj vidi na spletni strani.

4.3.1 Vpis v sistem

Uporabnik se vpiše v sistem s pomočjo uporabniškega imena in gesla. Če uporabnik še nima dostopa, mora za to prositi administratorja sistema, ki ga vpiše v sistem in ta mu posreduje geslo na e-poštni naslov. Če je geslo pozabljeno, lahko preko forme zahtevamo novo geslo, ki je prav tako posredovano na uporabnikovo e-pošto. Vpis v sistem se nahaja na podstrani spletne strani, ki se imenuje `/MyCMS`.



Slika 4.3: Vpis v sistem

4.3.2 Urejanje spletne strani

Uporabniki, ki imajo pravice za urejanje spletne strani v aplikaciji s pomočjo HTML značke `<iframe>` vidijo spletno stran. Zaradi varnostnih nastavitev nam brskalnik ne pusti dostopati do strani v `<iframe>` objektu, zato je nujno, da se aplikacija in spletna stran nahajata na isti domeni. Urejevalnik je vizualen, kar pomeni, da uporabnik vidi spletno stran in jo ureja preprosto s klikom na gumb "uredi", ki se nam pokaže ob prehodu miške skozi element, ki se lahko ureja. Ob kliku na gumb se nam prikaže modalno okno, v katerem se glede na vsebino prikaže forma za urejanje. Komunikacija med spletno stranjo in urejevalnikom poteka preko JavaScript dokumentov. Ob zaključku urejanja se spletna stran ne naloži ponovno, vendar se preko JavaScripta zamenja vsebina, tako uporabnik takoj vidi, če je zamenjana vsebina primerna in jo v ob napaki takoj ponovno popravi.

Slika 4.4: Urejevalnik spletne strani

Komunikacija spletne strani z urejevalnikom

Spletna stran komunicira z urejevalnikom s pomočjo JavaScripta. Ob kliku na gumb "uredi" v urejevalniku se na spletni strani sproži dogodek (event). Preko JavaScript sintakse dostopamo do nadrejenega okna, ki je v našem primeru urejevalnik. Urejevalniku sporočimo ime, id strani, id jezika, tip in velikost izbrane vsebine, kar dobimo preko parametrov izbranega elementa. Urejevalnik preko dobljenih parametrov prikaže modalno

okno, primerno za urejanje izbrane vsebine.

```
$('.editButton').live('click', function () {
    if (window.parent.document.activeElement.id === "page") { //Ce ima nadrejeno okno
         id page
        var width = 1;
        var height = 1;
        var par = $(this).closest('par');//Element z vsebino, ki jo zelimo urejati
        $('par[data-section-title]').removeClass("hover");
        var sectionTitle = par.attr('data-section-title'); //Ime vsebine
        var pageId = par.attr('data-page-id'); //Id strani
        var type = par.attr('data-section-type'); //Tip vsebine
        var languageId = $('meta[name=LanguageId]').attr("content"); //Id jezika
             spletne strani
        if (type === 3) {
            //V primeru urejanja slike pridobimo se sirino ter visino slike
            width = par.attr('data-section-width');
            height = par.attr('data-section-height');
        }

        //Sporocanje v urejevalnik
        window.parent.EditPageSection.popUp(sectionTitle, pageId, languageId, type,
             width, height);
    }
});
```

Algoritem 2: Klik na gumb 'uredi'

Prikaz modalnega okna za urejanje

Spletna stran preko JavaScripta pošlje urejevalniku ukaz, naj prikaže modalno okno, ki je primerno za vsebino. Preko parametrov dobimo informacijo, katero vsebino moramo prikazati v urejevalniku.

Zaporedna številka	Parameter	Opis
1	`title`	ime vsebine
2	`pageId`	`Id` strani
3	`languageId`	`Id` jezika strani
4	`type`	tip vsebine
5	`width`	širina slike
6	`height`	višina slike

Tabela 4.7: Parametri za prikaz modalnega okna z urejevalnikom

JavaScript funkcija pošlje parametre na strežnik, ta nam odgovori s HTML elementi, ki jih nato prikažemo v modalnem oknu. Akcija na strežniku se imenuje `EditPageSection` in se nahaja na kontrolerju `Page`.

```
popUp: function (title, pageId, languageId, type, width, height) {
    //Globalni parametri
    EditPageSection.sectionTitle = title;
    EditPageSection.sectionPageId = pageId;
    EditPageSection.sectionLanguageId = languageId;
    EditPageSection.sectionType = type;
    EditPageSection.pictureAspectRatio = width / height;

    //Parametri za pridobitev vsebine
    var parmeters = {
        title: title,
        pageId: pageId,
        languageId: languageId,
        sectionType: type
    };

    //Pretvorba v url paramtere
    var urlParameters = $.param(parmeters, true);

    setTimeout(function () {
        //Pridobitev podatkov s serverja
        EditPageSection.$detailsModal.load('/MyCms/Page/EditPageSection/' + "?" +
             urlParameters, '', function () {
            //Navadno besedilo
            if (EditPageSection.sectionType == SectionType.Text) {...}
            //Obogateno besedilo
            else if (EditPageSection.sectionType == SectionType.RichText) {...}
            //Slka
            else if (EditPageSection.sectionType == SectionType.Picture) {...}
            //Prikaz modalnega okna
            EditPageSection.$detailsModal.modal({
                backdrop: 'static',
                keyboard: false
            });
        });
    }, 500);
},
```

Algoritem 3: Funkcija za prikaz modalnega okna

Server preko parametrov naloži pravilno vsebino, ki v HTML obliki pošlje funkciji v JavaScript-u. JavaScript pa jo nam prikaže v modalnem oknu.

```
//Akcija za pridobitev vsebine na kotrolerju
//Parametri
//title - ime vsebine, pageId - id stran, languageId - id jezika, sectionType - tip
     vsebine
public ActionResult EditPageSection(string title, int pageId, int languageId, int
     sectionType)
{
    PageSectionDetailsModel model = new PageSectionDetailsModel();
    //Funkcija za nalaganje vsebine
    model.Load(title, pageId, languageId, sectionType);
    //Vrnemo HTML z vsebino
    return View("EditPageSection", model);
}

//Funkcija za nalaganje vsebine iz podatkovne baze v modelu
public bool Load(string title, int pageId, int langaugeId, int sectionType)
{
    //Nalaganje PageSection iz podatkovne baze
    PageSection pageSection = Db.PageSections.FirstOrDefault(x => x.Title == title
                                                                  && x.PageId == pageId
                                                                  && x.ItemTypeId ==
                                                                       sectionType);
    // Ce pageSection ni enak null nadaljujemo z branjem iz baze
    if (pageSection != null)
    {
        this.SectionTitle = pageSection.Title;
        this.Type = pageSection.ItemTypeId;
        this.SectionId = pageSection.ItemId;
        this.LanguageId = langaugeId;
        this.PageId = pageId;
        //Navadno besedilo
        if (this.Type == 1)
```

```
        {
            PageSectionText pageSectionText = Db.PageSectionTexts.FirstOrDefault(x =>
                x.Id == SectionId && x.LanguageId == langaugeId);
            this.ItemSectionId = pageSectionText.Id;
            this.Value = pageSectionText.Value;
        }
        //Obogateno besedilo
        else if (this.Type == 2)
        {
            PageSectionRichText pageSectionRichText = Db.PageSectionRichTexts.
                FirstOrDefault(x => x.Id == SectionId && x.LanguageId == langaugeId);
            this.ItemSectionId = pageSectionRichText.Id;
            this.Value = pageSectionRichText.Value;
        }
        //Slika
        else if (this.Type == 3)
        {
            PageSectionPicture pageSectionPicture = Db.PageSectionPictures.
                FirstOrDefault(x => x.Id == SectionId && x.LanguageId == LanguageId);
            this.ItemSectionId = pageSectionPicture.Id;
            this.Value = getPicturePath(pageSectionPicture.Name);
        }

        return true;
    }
    return false;
}
```

Algoritem 4: Nalaganje vsebine iz podatkovne baze

```
@using Nejc.Web.Minus.Models.Page
@model PageSectionDetailsModel
@{
    Layout = "~/Views/Shared/_ModalLayout.cshtml";
}
@section title{
    <h3>Urejanje</h3>
}
@section footer{
    <button id="save" type="button" class="btn green" data-dismiss="modal">Shrani
        </button>
    <button type="button" class="btn red" data-dismiss="modal">Zapri </button>
}
@if (Model.Type == 1)
{
    <textarea id="textArea" maxlength="50" style="resize:none; height:22px; width:
        98%;">@Model.Value</textarea>
}
else if (Model.Type == 2)
{
    <form class="form-horizontal" action="#">
        <div class="control-group">
            <div class="controls">
                <textarea id="textArea" style="width: 98%; margin: 0;" class="span12
                    ckeditor m-wrap" rows="12">
                    @Model.Value
                </textarea>
            </div>
        </div>
    </form>
}
else if (Model.Type == 3)
{
    <div class="container" style="width:500px; height: 250px;" >
        <img id="image" src="@Model.Value">
    </div>
    <label class="btn btn-upload" for="inputImage" title="Upload image file"
           style="margin: 15px 0px 0px 15px; width: 500px; padding:0;">
        <input class="sr-only" id="inputImage" name="file" type="file" accept="image/*
            ">
    </label>
    <canvas id="canvas" style="height:0px; width:0px;"></canvas>
}
```

Algoritem 5: HTML koda za prikaz modalnega okna

Urejanje navadnega besedila

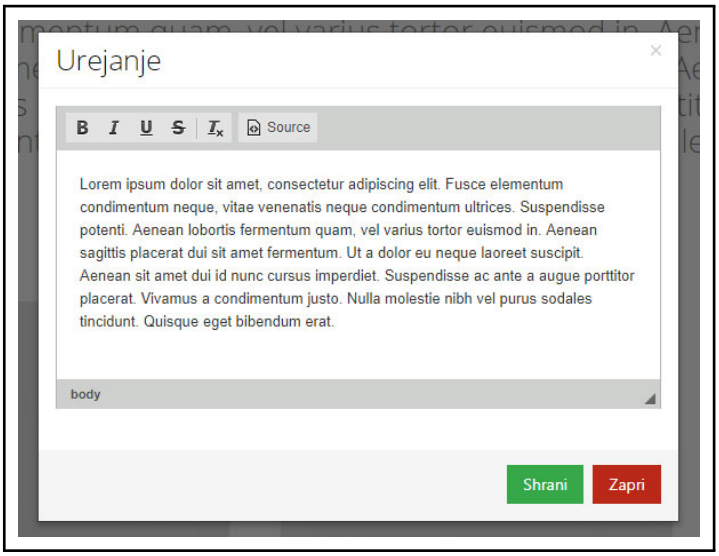
Navadna besedila se uporablja za naslove in krajše stavke, ki nimajo dodatnih stilov. Ob kliku na gumb za urejanje se na strežnik pošlje zahteva za pridobitev vsebine tipa 1 - navadno besedilo. Na strežniku se najprej po imenu, tipu in id strani naloži element iz tabele `PageSection`, nato se glede na tip naloži vsebino iz tabele `PageSectionText`. Vsebino, ki je bila naložena, nato glede na tip prikažemo v modalnem oknu. Pri navadnem besedilu je to HTML element `<textarea>`, ki nima posebnih nastavitev.

Slika 4.5: Modalno okno za urejanje navadnega besedila

Urejanje obogatenega besedila

Obogatena besedila se uporablja za besedila, ki so daljša od nekaj besed in jih želimo poljubno oblikovati. Ob kliku na gumb za urejanje se na strežnik pošlje zahteva za pridobitev vsebine tipa 2 - obogateno besedilo. Na strežniku se najprej po imenu, tipu in id strani naloži element iz tabele `PageSection`, nato se glede na tip naloži vsebino iz tabele `PageSectionRichText`. Vsebino, ki je bila naložena nato glede na tip, prikažemo v modalnem oknu. Pri obogatenem besedilu je to HTML element `<textarea>`, ki pa potrebuje dodatno inicializacijo urejevalnika v JavaScriptu. Za pomoč uporabimo CKEDITOR[1], ki nam oblikuje besedilo po naših željah. Besedilo lahko stilno uredimo. Imamo standardne stile, kot so: krepko, ležeče, podčrtano in prečrtano. Lahko tudi urejamo samo HTML kodo besedila.

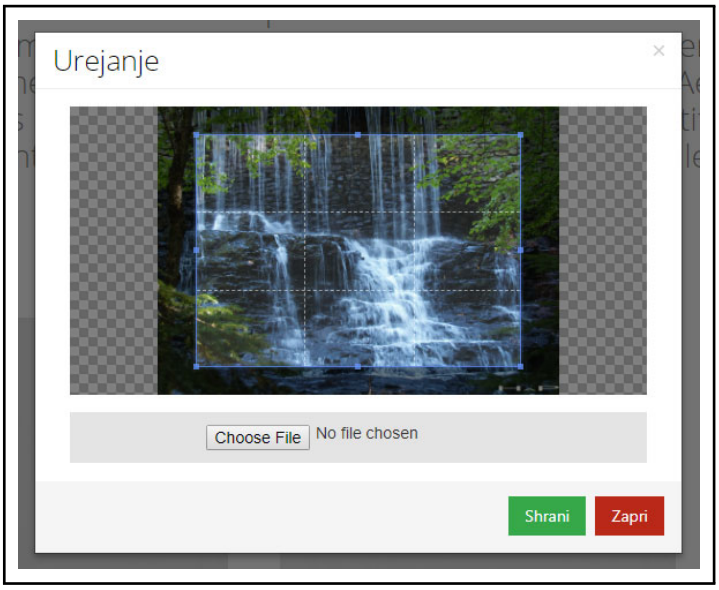
[1] https://ckeditor.com/



Slika 4.6: Modalno okno za urejanje obogatenega besedila

Urejanje slike

Ob kliku na gumb za urejanje se na strežnik pošlje zahteva za pridobitev vsebine tipa 3 - slika. Na strežniku se najprej po imenu, tipu in id strani naloži element iz tabele `PageSection`, nato se glede na tip naloži vsebino iz tabele `PageSectionPicture`. Sliko nato prikažemo v modalnem oknu, ki vsebuje urejevalnik. Poleg slike prikažemo še gumb za nalaganje nove slike iz računalnika. Preko JavaScript inicializiramo urejevalnik, ki nam v `<canvas>` elementu prikaže našo sliko. Slika ima razmerje nastavljeno tako, kot ga potrebujemo za prikaz na spletni strani. Urejevalnik nam omogoča nalaganje slike preko gumba za nalaganje. Sliko imamo možnost obrezati na velikost oz. izrezati določen del slike, ki ga želimo prikazati, za kar nam služi funkcija cropper.



Slika 4.7: Modalno okno za urejanje slik

4.3.3 Shranjevanje in posodabljanje vsebine

Za shranjevanje sprememb kliknemo na gumb shrani. Ob kliku na gumb preko JavaScript in Ajax zahteve pošljemo podatke, ki smo jih zajeli glede na tip vsebine, na strežnik preko akcije `UpdatePageSection`.

V primeru urejanja navadnega besedila vrnemo vpisano besedilo brez oblikovanja. Če imamo obogateno besedilo, pridobimo podatke iz urejevalnika besedila in ga skupaj z oblikovanjem pošljemo na strežnik, če pa imamo sliko pa nam funkcija pošlje nazaj podatke o sliki v Base64 obliki, ki je binarni zapis slike.

```
EditPageSection.$detailsModal = $('#details-modal');
$('#save').live('click', function () {
    var pictureName = "";
    var value = "";
    if (EditPageSection.sectionType == SectionType.Text) {
        value = $('#textArea').val();
    }
    else if (EditPageSection.sectionType == SectionType.RichText) {
        value = CKEDITOR.instances.textArea.getData();
        var length = value.length;
        value = value.substring(3, length - 5);
    }
    else if (EditPageSection.sectionType == SectionType.Picture) {
        pictureName = $('#inputImage').val();
        var type = "image/png";
        var quality = 1.0;

        var canvas = document.getElementById("canvas"),
            data = EditPageSection.pictureData,
            pictureURL = "",
            context;

        var options = $.extend({
            width: data.width,
            height: data.height
        }, options);

        canvas.width = options.width;
        canvas.height = options.height;
        context = canvas.getContext("2d");

        context.drawImage(EditPageSection.image, data.x, data.y, data.width, data.
            height, 0, 0, options.width, options.height);
        pictureURL = canvas.toDataURL(type, quality);
        value = pictureURL;
        pictureName = $('#inputImage').val();
    }

    var params = {
        SectionTitle: EditPageSection.sectionTitle,
        Value: value,
        LanguageId: EditPageSection.sectionLanguageId,
        PageId: EditPageSection.sectionPageId,
        Type: EditPageSection.sectionType,
        PictureName: pictureName
    };

    $.post("/MyCms/Page/UpdatePageSection/", params, function (data) {});

    if (EditPageSection.sectionType == SectionType.Text || EditPageSection.sectionType
          == SectionType.RichText) {
        document.getElementById('page').contentWindow.setText(EditPageSection.
            sectionTitle, value);
    }
    else {
        document.getElementById('page').contentWindow.setPicture(EditPageSection.
            sectionTitle, value);
    }
});
```

Algoritem 6: Shranjevanje vsebine v JavaScript

Na strežniku najprej naložimo entiteto iz tabele `PageSection`, nato pa preko tipa vsebine naložimo še druge podatke.

Za navadno besedilo naložimo vsebino iz tabele `PageSectionText`. Za obogateno besedilo pa naložimo iz tabele `PageSectionRichText`. Nato za oba tipa zamenjamo vrednost polja `Value` z novo vrednostjo, ki smo jo dobili preko JavaScripta.

Pri urejanju slike naložimo vsebino iz tabele `PageSectionPicture`. Nato iz imena

nove slike pridobimo končnico datoteke. če se datoteka ni spremenila, vzamemo končnico iz baze. Pomembna je zaradi pravilnega shranjevanja slike. Sliko v zapisu Base64 shranimo v datoteko, ki ji določimo ime. Nato to ime zapišemo v podatkovno bazo v polje `Name`, polja `GuId` in `SharedGuId` vsebujeta ime slike brez končnice.

Po zaključku shranjevanja nam strežnik odgovori z uspešnostjo zahteve. Staro vsebino elementa preko JavaScripta zamenjamo in tako upravljavec spletne strani takoj vidi spremembo, ki jo je opravil. Prav tako spremembo vidijo uporabniki, ki bodo od takrat naprej obiskali spletno stran.

```
private string getPicturePath(string name)
{
    string path = "\\Content\\Images\\img\\" + name;
    return path;
}

public bool Update()
{
    //Iz baze PageSection pridobimo pravi PageSection
    PageSection pageSection = Db.PageSections.FirstOrDefault(x => x.Title ==
        SectionTitle && x.PageId == PageId && x.ItemTypeId == Type);
    //Ce je pageSection.ItemTypeId enak 1 pomeni da gre za navaden text zato podatke
        zapisemo v PageSectionText tabelo

    if (pageSection.ItemTypeId == 1)
    {
        PageSectionText pageSectionText = Db.PageSectionTexts.FirstOrDefault(x => x.
            SectionId == pageSection.Id && x.LanguageId == LanguageId);
        pageSectionText.Value = this.Value;
    }
    else if (pageSection.ItemTypeId == 2)
    {
        PageSectionRichText pageSectionRichText = Db.PageSectionRichTexts.
            FirstOrDefault(x => x.SectionId == pageSection.Id && x.LanguageId ==
            LanguageId);
        pageSectionRichText.Value = this.Value;
    }
    else if (pageSection.ItemTypeId == 3)
    {
        PageSectionPicture pageSectionPicutre = Db.PageSectionPictures.FirstOrDefault(
            x => x.SectionId == pageSection.Id && x.LanguageId == LanguageId);

        this.PictureExtension = ".jpg";

        if (!string.IsNullOrEmpty(this.PictureName))
        {
            this.PictureExtension = Path.GetExtension(this.PictureName);
        }
        else if (pageSectionPicutre != null)
        {
            this.PictureExtension = Path.GetExtension(getPicturePath(
                pageSectionPicutre.GuId));
        }

        string base64 = this.Value.Substring(this.Value.IndexOf(',') + 1);
        base64 = base64.Trim('\0');
        pageSectionPicutre.SharedGuId = SaveImage(base64);
        pageSectionPicutre.GuId = pageSectionPicutre.SharedGuId;
        pageSectionPicutre.Name = pageSectionPicutre.GuId + this.PictureExtension;
    }
    Db.SubmitChanges();
    return true;
}

public string SaveImage(string base64String)
{
    var bytes = Convert.FromBase64String(base64String);
    string GuId = Guid.NewGuid().ToString();
    string filePath = AppSettings.ImageFolder + "/" + GuId + this.PictureExtension;
    using (var imageFile = new FileStream(filePath, FileMode.Create))
    {
        imageFile.Write(bytes, 0, bytes.Length);
        imageFile.Flush();
        return GuId;
    }
}
```

Algoritem 7: Nalaganje vsebine iz podatkovne baze

5 Zaključek

V diplomskem delu smo predstavili problem in potrebo po sistemu za urejanje spletne vsebine, ki bi omogočal lažje ter hitrejše upravljanje spletne vsebine. Tako smo predstavili nekaj najbolj znanih sistemov, njihove prednost in slabosti. Nato smo pojasnili osnove uporabljenih tehnologij za razvoj sistema. Po pregledu tehnologij smo se odločili, da projekt izpeljemo na podlagi C# programskega jezika v MVC arhitekturi.

Naročnik skupaj s spletno stranjo dobi tudi možnost urejanja vsebine, in to mu omogoča lažje ter predvsem hitrejše delo. Ker ni nepotrebne komunikacije z razvijalcem, lahko takoj uredi vsebino spletne strani sam.

Cilj diplomskega dela je bil analizirati druge sisteme, ki se uporabljajo za urejanje spletne vsebine in razvoj lastnega sistema. Ugotovili smo, da vizualnega urejevalnika ni v odprtokodni rešitvi, zato smo se lotili izdelave. S tem smo dosegli cilj diplomskega dela, ki pa dopušča nadaljnji razvoj glede na potrebe naročnika.

5.1 Napotki za nadaljnji razvoj

V diplomski nalogi smo se lotili problema izdelave vizualnega urejevalnika spletne vsebine, ki smo ga tudi uspešno rešili in realizirali. Za nadaljnji razvoj nam ostane še dodelava varnosti, ki jo je potrebno redno vzdrževati. Pri naročnikih se pojavi tudi želja po objavi raznih galerij, blogov, projektov itd. V tem segmentu je sistem zelo fleksibilen in nam omogoča izdelavo po potrebi stranke.

Literatura

[1] Margaret Rouse: What is content managemenet system
https://searchcontentmanagement.techtarget.com/definition/content-management-system-CMS

[2] Tom Ewer: 14 Surprising Statistics About WordPress Usage
https://managewp.com/blog/14-surprising-statistics-about-wordpress-usage

[3] Wikipedia: WordPress
http://en.wikipedia.org/wiki/WordPress

[4] Drupal: What is Drupal?
https://www.drupal.com/product/web-content-management

[5] Rockettheme: Joomla
https://rockettheme.com/docs/joomla/platform/INDEX.md

[6] W3Schools: SQL
https://www.w3schools.com/sql/sql_intro.asp

[7] Wikipedia: SQL Management Studio
https://en.wikipedia.org/wiki/SQL_Server_Management_Studio

[8] Svetlin Nakov & Co. *FUNDAMENTALS OF COMPUTER PROGRAMMING WITH C Sharp*, 18-20, 924-925, 2013.

[9] JQuery.com: jQuery
https://jquery.com

[10] Wikipedia: jQuery
https://en.wikipedia.org/wiki/JQuery

[11] Wikipedia: JavaScript
https://en.wikipedia.org/wiki/JavaScript

[12] S. Suehring, J. Valade, *PHP, MySQL, JavaScript & HTML5 All-in-One For Dummies*, 89-90, 121-122, 187-188, 2013.

[13] Mozilla.org: Getting Started with Ajax
https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started

[14] W3Schools: HTML
https://www.w3schools.com/html/html_intro.asp

[15] Wikipedia: CSS
https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[16] Marc Wilson: What is IIS? A Basic Tutorial of the Windows Web Server
https://www.pcwdld.com/what-is-iis

[17] Microsoft: Visual Studio
https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2017

[18] Mozilla: MVC arhictecture
https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture