Sustav za upravljanje web sadržajem: gotov proizvod ili samostalna izrada?

Kmetič, Stela

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, University of Zagreb, Faculty of Humanities and Social Sciences / Sveučilište u Zagrebu, Filozofski fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:131:094621

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. god. 2019./2020.

Stela Kmetič

Sustav za upravljanje web sadržajem: gotov proizvod ili samostalna izrada?

Završni rad

Mentor: dr.sc. Kristina Kocijan, izv. prof.

Zagreb 2020.

Izjava o akademskoj čestitosti

Izjavljujem i svojim potpisom potvrđujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

_____

(potpis)

Sadržaj

1. Uvod

Web stranice su posjetitelju prvi, a ponekad i posljednji, dojam koji će dobiti o određenoj tvrtki. U današnjoj sferi mnoštva opcija i kratkoročne pažnje, bitno je da taj prvi dojam bude pozitivan. Svaka korporativna web stranica primarno je fokusirana na svoj sadržaj, jer će čak i zastarjeli dizajn biti oprošten pored kvalitetnog sadržaja i dobro organizirane i prohodne web stranice (Holst, 2013). Iduće pitanje koje se onda nameće je: „*Što se podrazumijeva pod kvalitetnim sadržajem?*“. Odgovora na ovo pitanje može biti nebrojeno mnogo, no jedan od odgovora je univerzalan: ***Bitna stavka kvalitetnog web sadržaja je njegova dinamičnost i aktualnost***.

Jedan dobar primjer ovoga je situacija kroz koju smo prošli 2020. Uslijed pandemije mnogi dućani nisu radili ili su imali promijenjeno radno vrijeme. Kada bi netko za određeni dućan na njihovoj web stranici htio provjeriti novo radno vrijeme i ne bi pronašao tražene podatke, vjerojatno bi otišao konkurenciji kod koje je mogao sa sigurnošću utvrditi radno vrijeme.

No, kako se postiže aktualnost sadržaja na web stranici? Kratak odgovor bi bio: **sustavima za upravljanje web sadržajem** (engl. *Content Management System*). Bez modernih sustava za upravljanje web sadržajem, cijena aktualnosti bila bi visoka jer bi svaka web stranica zahtijevala prisutnost web programera koji bi kroz sam kod mijenjao sadržaj svaki puta kada bi kakva promjena bila potrebna. Moderni sustavi omogućavaju zaobilaženje programera i klijentu daju pristupačno vizualno sučelje u kojem je aktualizacija brza i jednostavna.

Upravo ti sustavi fokus su ovoga rada, te će se unutar njega dotaknuti povijesti sustava i tehnologija za upravljanje web sadržajem, prednosti i mana već gotovih, upakiranih sustava u usporedbi s onima u samostalnoj izradi, te same arhitekture i vrsta istih tih sustava.

2. Povijest sustava za izradu web sadržaja i popratnih tehnologija

Sustavi za izradu web sadržaja oslanjaju se na već etablirane tehnologije skriptiranja sa strane poslužitelja (tehnologije koje pozivaju poslužitelj kada klijent zatraži ili unese određene podatke na web stranicu), odnosno na tehnologije koje omogućuju dinamičnost i interaktivnost između klijenta i poslužitelja. Ovo poglavlje nudi povijesni pregled tehnologija za skriptiranje sa strane poslužitelja i prvih sustava za izradu web sadržaja u vidu razvitka weba od weba 1.0 pa sve do danas.

2.1. Web 1.0

Povijest sustava za izradu web sadržaja započinje 1989. kada je Tim Berners-Lee predložio HTML (engl. *Hypertext Markup Language*) i napisao prvi web preglednik 1990. godine (World Wide Web Consortium, 2020). Prve web stranice bile su jednostavne HTML tekstualne datoteke, te se koristilo FTP (engl. *File Transfer Protocol*) kako bi ih se postavilo na poslužitelj, odnosno kako bi se aktualiziralo web stranicu (Longman, 1998). Ovakve web stranice spadaju u web 1.0 eru, odnosno prvu fazu kada su sadržaj na internetu karakterizirale jednostavne statične web stranice (Nauk, Shivalingaiah, 2008).

Začetkom web preglednika pojavila se i prva inačica sustava za izradu web sadržaja u obliku ***Server Side Includes*** (SSI). SSI je skriptni jezik na strani poslužitelja koji omogućava:

- prikazivanje i stvaranje varijabli,
- razdvajanje tj. inkluziju sadržaja u više datoteka, npr. razdvajanje zaglavlja (engl. header) i podnožja (engl. footer) od ostatka sadržaja,
- prikazivanje informacija poput datuma ili naziva datoteke,
- izvršavanje programa ili naredbi,
- uporabu jednostavnih uvjetnih izraza (The Apache Software Foundation, 2020).

SSI je i danas prisutan na webu, primarno za jednostavno dodavanje informacija i datoteka od kuda mu i potječe ime (engl. *includes* = uključuje), no u području generiranja dinamičkog sadržaja zamijenili su ga brži jezici koji obuhvaćaju veću sferu mogućnosti (Lee, 2002).

U otprilike isto doba kada se pojavio SSI, izrađen je i ***Common Gateway Interface*** (CGI). 1993., tim u sklopu *National Center for Supercomputing Applications* napisao je specifikacije za pozivanje izvršnih naredbi kroz web stranice, koje su do 1997. bile objedinjene i dorađene pod nazivom CGI (Connolly, 2020).

CGI za svoje je doba bio revolucionaran, što je i vidljivo iz literature koja ga je popratila:

„*Dok putujete World Wide Webom, susrest ćete stranice koje će vas navesti da se zapitate „Kako su to napravili?“. Te stranice mogu se sastojati, između ostalog, od formulara koji traže registraciju ili povratne informacije, interaktivnih mapa slika, brojača koji prikazuju broj posjeta i funkcija koje vam omogućavaju pretraživanje baze za određene podatke. U većini slučajeva, otkrit ćete da su te web stranice izrađene u Common Gateway Interfaceu.*“ (Gundavaram, 1996:5)[1]

CGI se smatra začetnikom dinamičnih web stranica jer dozvoljava komunikaciju između poslužitelja i web stranice u stvarnom vremenu, dok je SSI komunikaciju dozvoljavao samo kada se stranica sa poslužitelja slala primatelju (Lemay, Moncur, 1996).

On funkcionira slično kao i njegovi nasljednici mod_perl i PHP, tj. putem POST i GET zahtjeva šalje, odnosno zahtijeva, određene informacije od poslužitelja. CGI datoteka na poslužitelju obrađuje zahtjev i šalje zatražene podatke.

1997. ColdFusion razvio je jezik za skriptiranje pod nazivom **ColdFusion Markup Language** (CFML). CFML je nadogradnja na HTML koja dodaje pristup bazama podataka, uvjetne operatere, funkcije formatiranja i druge elemente u svrhu stvaranja web aplikacije (Maiano, 2019). Ovaj jezik uspio je uvelike popularizirati skriptiranje sa strane poslužitelja, a koristi se i dan danas u iteracijama Adobea[2], Luceeja[3] i mnogih drugih.

Perl, Python, CGI i CFML začetnici su skriptiranja sa strane poslužitelja kojima su se kasnije pridružili **Personal Home Page** (PHP), **Active Server Pages** (ASP) i **JavaServer Pages** (JSP).

Doba Web 1.0 službeno je završilo 1997. uvođenjem Document Object Modela (DOM). DOM definira logičku strukturu dokumenta i način na koji se dokumentu pristupa, te kako ga se

[1] Izvorni citat: „*As you traverse the vast frontier of the World Wide Web, you will come across documents that make you wonder, "How did they do this?" These documents could consist of, among other things, forms that ask for feedback or registration information, imagemaps that allow you to click on various parts of the image, counters that display the number of users that accessed the document, and utilities that allow you to search databases for particular information. In most cases, you'll find that these effects were achieved using the Common Gateway Interface, commonly known as CGI.*“

[2] https://www.adobe.com/products/coldfusion-family.html

[3] https://lucee.org/

manipulira (Robie, 2020). On se smatra aplikacijskim programskim sučeljem (engl. *Application programming interface* - API) za HTML i XML dokumente jer dozvoljava manipulaciju nad specifičnim elementima HTML-a, npr. mijenjanje stila za „body“ oznaku.

2.1.1. PHP

PHP-u je idejni začetnik bio Rasmus Lerdorf, a prva iteracija PHP-a bio je set CGI naredbi napisanih u C jeziku. Lerdorf je originalno taj set naredbi koristio kako bi pratio broj posjeta web stranici koja je sadržavala njegov životopis, od kuda je i došao naziv *Personal Home Page Tools*, odnosno alati za osobnu web stranicu. PHP je s vremenom dodavao nove funkcije, uključujući i mogućnosti interakcije s bazom podataka, tako omogućavajući izradu dinamičkih web aplikacija. Ledorf je sav svoj kod napravio javnim i pozivao na uključenje programerske zajednice u daljnji razvitak PHP-a kao zasebnog jezika. Tek je 1996. PHP (pod nazivom PHP/FI – Forms Interpreter) zaživio kao zasebni jezik a ne samo iteracija CGI naredbi (PHP, 2020).

Ova verzija PHP-a pisala se unutar HTML datoteke u komentarima, što je vidljivo iz sljedećeg primjera:

```
<!--include /text/header.html-->

<!--getenv HTTP_USER_AGENT-->
<!--ifsubstr $exec_result Mozilla-->
Hey, you are using Netscape!<p>
<!--endif-->
```

PHP/FI uskoro je prerastao u PHP 2.0, kojega je je potom naslijedio PHP 3.0, prva verzija PHP-a slična onoj koju danas koristimo. PHP 3.0 zaživio je zahvaljujući Andiju Gutmansu i Zeevu Suraskiju, dvojci studenata iz Tel Aviva. Gutman i Suraski skupa su radili na fakultetskom projektu u koji su htjeli implementirati PHP, no tadašnja verzija PHP-a bila je preograničena za njihove potrebe. U suradnji s Lerdorfom preradili su parser za PHP i nadjenuli mu ime PHP:*Hypertext Preprocessor*. PHP 3.0 omogućavao je rad s više baza podataka, protokola i API-a. Modularnost jezika privukla je mnoge programere koji su za jezik pisali vlastite module, što je pridonijelo masovnom uspjehu tadašnje inačice PHP-a – čak 10% web stranica 1998. koristilo je PHP.

Danas radimo s PHP 7.0, do sada, naravno, najboljom i najbržom inačicom PHP-a. Čak 79% stranica danas koristi PHP, što ga čini daleko najzastupljenijim skriptnim jezikom na web tržištu (W3Techs, 2020). PHP u svoje korisnike ubraja i divove poput Facebooka, Wikipedie, Yahooa, Photobucketa i Wordpressa, danas najvećeg sustava za izradu web sadržaja.

2.1.2. ASP

Prva inačica ASP-a bilo je Microsoftovo aplikacijsko programsko sučelje za internetski poslužitelj (engl. *Internet Server Application Programming Interface* - ISAPI) 1995. godine (Weissinger, 1999). ISAPI je bio stvoren kao alternativa CGI-u koja je adresirala jednu od najvećih ograničenja CGI-a: svaki puta kada bi klijent zatražio pokretanje neke CGI aplikacije, poslužitelj bi otvorio novu instancu CGI-a. Ovakav pristup značajno je teretio poslužitelj, a Microsoft je problem riješio oslanjajući se na DLL (*Dynamic Link Libraries*). Svaka ISAPI aplikacija bila bi napravljena u obliku jednog DLL-a koji bi bio učitan u istu memoriju koju koristi server, te bi tamo i ostao odgovarajući na upite sve dok ne bi bio eksplicitno maknut iz memorije.

ASP je nastao iz ISAPI 2.0 1996. ASP se u svojoj originalnoj iteraciji sastojao od jedne DLL datoteke velike samo 300Kb, te kada god bi klijent zatražio datoteku s asp ekstenzijom, zahtjev bi bio poslan na obradu .asp datoteci koja bi nakon obrade podatke proslijedila serveru koji bi ga dalje injektirao u HTML koji šalje u preglednik.

Zadnja verzija originalnog ASP-a bila je uključena u Windows Server 2008, te je službeno prestala primati podršku od Microsofta 14.1.2020 (Microsoft, 2020). Originalni ASP danas je zamijenio ASP.net, koji prema broju korisnika stoji odmah iza PHP-a s respektabilnih 10.1% tržišta (W3Techs, 2020).

2.2. Web 2.0

Eru web 2.0 primarno karakterizira interaktivnost i dinamičnost, te mu je zato i nadjenuto ime „*read-write*“ perioda interneta (Nauk, Shivalingaiah, 2008). Sam izraz web 2.0 prvi puta je iskorišten u listopadu 2004. na O'Reilly Media Web 2.0 konferenciji, a nastao kao rezultat saznanja o novim mogućnostima weba (Graham, 2005). O'Reilly je tada webu 2.0 dao sljedeća obilježja:

- Internet kao platforma,
- arhitektura participacije,
- inovacija i nezavisni programeri,
- otvorenost podataka.

Pojava DOM-a omogućila je stvaranje prvih uistinu dinamičnih web stranica: koristeći Asinkroni JavaScript i XML (**Ajax**) po prvi puta je bilo moguće primati i slati podatke bez ponovnog učitavanja web stranice. Izraz Ajax osmislio je Jesse James Garrett u članku „*Ajax: A New Approach to Web Applications*“ 2005. godine. U samome uvodu, Garret (2005.) navodi kako je od samog začetka interneta interaktivnost slična onoj u desktop aplikacijama bila nedostižna u web okružju, no na pomolu je bila velika promjena. Google Suggest i Google Maps primjer su prvih web aplikacija koje su bile uistinu interaktivne bez potrebe za ponovnim učitavanjem stranice. Google Suggest gotovo je trenutačno nadopunjavao upite koje bi korisnik utipkavao, dok je u Google Mapsu bilo moguće zumiranje i beskrajno kretanje po mapi (Garrett, 2005).

Garret (2005.) navodi da Ajax nije zasebna tehnologija već naziv za kombinaciju više tehnologija:

- prezentacija korištenjem XHTML-a (Extensible HyperText Markup Language) i CSS-a (Cascading Style Sheets),
- dinamične interakcije koristeći DOM,
- izmjena i manipulacija podataka korištenjem XML-a (Extensible Markup Language) i XSLT-a (Extensible Stylesheet Language Transformations),
- asinkrono zaprimanje podataka kroz XMLHTTP zahtjeve,
- i JavaScript kao poveznica svih ostalih tehnologija.

Novonastala dinamičnost pridonijela je interaktivnosti na webu, pa se ovu eru naziva i erom socijalnog ili participativnog weba. Količina sadržaja stvorenog od strane korisnika koji nisu nužno sami posjedovali web stranice bitno se povećala – ovo je bila era komentara i popularizacije društvenih medija.

Od Friendstera do Myspacea, količina korisnika je eksplodirala, a za to su sve bili zaslužni sustavi za upravljanje web sadržajem koji su u pozadini radili kako bi prikazali unesene informacije. To su bili **monolitski** ili **tradicionalni** sustavi za upravljanje web sadržajem čija će arhitektura biti objašnjena u zasebnom poglavlju.

Jednim od prvih monolitskih sustava smatra se FileNet, osnovan 1982., a izdan kao potpuno funkcionalni sustav za izradu web sadržaja 1995. godine (Carpenter, 2011). FileNet su uskoro slijedili i drugi sustavi poput Vignette, Documentum, FutureTense, EpiServer i Ipso, te se može reći da je sredina 90-ih bila svojevrsni „boom“ razvitka sustava za upravljanje web sadržajem.

U samim počecima web 2.0 ere, počeli su se pojavljivati sustavi otvorenog koda kao Zend, PHP-Nuke, OpenCMS, WordPress, Plone, Drupal i Joomla, od kojih su neki veoma brzo evoluirali u platforme za izradu web sadržaja u obliku u kojem ih poznajemo i danas. 2003. WordPress dodaje dizajnerske elemente i na tržištu se pojavljuje SquareSpace. 2006. pojavljuju se Weebly i Wix.

2.3. Web 3.0

Pojam web 3.0 prvi je puta osvanuo u New York Timesu 2006. u članku pod naslovom “*A 'more revolutionary' Web*” (Shannon, 2006). U njemu su prenesene osnovne karakteristike weba 3.0 koje je definirao Tim Berners Lee: 3d grafike, semantički, odnosno inteligentni web, multi-kanalnost i povezanost.

Čini se da su se njegova predviđanja i ostvarila. Danas živimo u svijetu inteligentnih algoritama, povezanosti sadržaja diljem platformi, 3D i skalabilnih vektorskih grafika, a polako razvijamo i umjetnu inteligenciju.

Povezanost sadržaja diljem platformi u velikom dijelu možemo zahvaliti razvitku sustava za izradu web sadržaja. Dok nam tradicionalni sustavi nisu omogućavali objavljivanje na više platformi istovremeno (npr. socijalni mediji, aplikacija, web stranica), novi sustavi nam to omogućuju, usput pridonoseći brzini učitavanja sadržaja, sigurnosti poslužitelja i skalabilnosti sustava.

Ti novi sustavi zovu se **raspareni** i **bezglavi** sustavi. Ovi sustavi nemaju mogućnost trenutačne i jednostavne instalacije poput danas najpoznatijih Wordpressa i Drupala, već je za njihovu instalaciju nezaobilazno poznavanje programskih jezika. Oni moraju bar djelomično biti sustavi u vlastitoj izradi.

3. Arhitektura sustava za upravljanje web sadržajem

U suštini, sustavi za izradu web sadržaja sastoje se od 2 komponente: aplikacije za upravljanje sadržajem i aplikacije za dostavljanje sadržaja. Aplikacija za upravljanje sadržajem omogućava administraciju nad korisnicima kako bi oni mogli stvarati, uređivati i micati sadržaj sa stranice, a sadrži i aplikaciju za dizajn koja ne zahtijeva poznavanje HTML-a, CSS-a ili programskih jezika, tako omogućujući rad bez programera. Aplikacija za dostavljanje sadržaja uzima preinake napravljene u aplikaciji za upravljanje web sadržajem i radi tražene preinake na web stranici.

Prema implementaciji te dvije komponente, razlikujemo **uparene**, **rasparene** i **bezglave** sustave.

3.1. Upareni ili tradicionalni sustavi za upravljanje web sadržajem

Upareni sustav za upravljanje web sadržajem najzastupljeniji su sustavi na tržištu, dobrim dijelom zahvaljujući tome što se na njih oslanjaju danas najpoznatije platforme za izradu web sadržaja poput Wordpressa, Drupala, Joomle, Squarespacea, Wixa itd.

U takvim sustavima, frontend (dio sustava koji korisnik direktno koristi) i backend (dio sustava s kojim korisnik direktno nema dodir) blisko su povezani: programeri i krajnji korisnici u suštini koriste isti sustav, lociran na istom mjestu.

Npr. programer pohranjuje kod za funkcionalnost web stranice i samog sustava (backend) u istu mapu na poslužitelju. Korisnik u frontendu pristupa istom tom sustavu, na istoj lokaciji, samo što umjesto koda koristi grafičko sučelje sustava kako bi napravio promjene. I sustav i frontend koriste istu arhitekturu, odnosno isti programski jezik.

Upareni sustav za upravljanje web sadržajem sastoji se od sljedećih komponenti:

- lokalna baza podataka za pohranu sadržaja (backend),
- sustav za upravljanje sadržajem na backendu (za stvaranje i aktualiziranje sadržaja),
- aplikacija u kojoj dizajneri mogu stvarati i primjenjivati sheme dizajna (backend)
- frontend koji prikazuje sadržaj na web stranici (Bateman, 2020).

3.1.1. Prednosti

1.) **Integrirani frontend i backend**

Iako ujedno i nedostatak, integracijom frontenda i backenda omogućuje se trenutačna primjena promjena na web stranicu, kao i pregled promjena prije finalne objave (preview). Još jedna bitna značajka koju integriranost omogućava je primjena backend modula jednim klikom. U npr. Wordpressu, pronalazimo razne opcije za funkcionalnost backenda koje možemo trenutačno instalirati i koristiti što nije moguće u rasparenim sustavima kojima su frontend i backend razdvojeni.

2.) **Lakoća instalacije**

Svi danas najpoznatiji sustavi za upravljanje sadržajem oslanjaju se na ovaj sustav, čak i Wordpress, danas najzastupljeniji i najpopularniji takav sustav (W3Techs, 2020). Wordpress svoju popularnost može zahvaliti nizu faktoru od kojih su cijena od 0kn, pristupačnost sučelja za stvaranje web stranica i lakoća instalacije na vrhu liste. On, kao i drugi slični sustavi poput Drupala, Joomle, Wixa i Squarespacea, za kompletnu instalaciju traže samo raspakiravanje zip datoteke u mapu domene i jednu bazu podataka u koju će moći pohranjivati sadržaj koji će korisnik stvarati. Ovakav proces instalacije netko tko želi samo npr. blog, može i samostalno odraditi uz pomoć nebrojenih videa i članaka koji se na internetu nalaze zahvaljujući popularnosti ovih platformi.

3.) **Uravnoteženost**

U uparenom sustavu za upravljanje web sadržajem postoji idealna uravnoteženost u kontroli nad sadržajem. Ovakvi sustavi daju veću kontrolu nad izgledom pojedinog članka i same web stranice klijentu, dok bezglavi sustavi bitno smanjuju opseg sadržaja nad kojim klijent ima kontrolu.

3.1.2. Nedostatci

1.) Integrirani frontend i backend

Iako klijentu omogućava veći opseg mogućnosti prilikom izrade sadržaja, integriranost frontenda i backenda lako može uzrokovati pad cijelog sustava. Jedna greška u kodu, gdje god se nalazila, utjecat će na cijeli sustav, a ukoliko dođe do sigurnosnog propusta, napadač će moći pristupiti lokalnoj bazi podataka, odnosno cjelokupnom sadržaju web stranice, ali i lozinkama koje su ondje pohranjene.

Pošto su i frontend i backend povezani, bilokakav rad na backendu direktno utječe na funkcionalnost frontenda. S takvom arhitekturom bit će više slučajeva u kojima je stranica nedostupna radi radova na backend arhitekturi.

2.) Neskalabilnost

Rješenje za povećani promet na web stranici u tradicionalnom sustavu je povećanje broja servera koji vrte isti taj sustav, no to rješenje nije najefikasnije. Druge verzije sustava za upravljanje web sadržajem nude bolja rješenja koja ne zahtijevaju preslikavanje cijelog sustava već mogu na druge načine razdijeliti zahtjeve koje server zaprima.

Također, što ako želimo sadržaj predstaviti na više kanala? Danas svaki sadržaj profesionalnog web projekta treba biti kompatibilan sa cijelim nizom uređaja i platformi: kompjuterima, laptopima, tabletima, mobitelima, moblinim/tablet aplikacijama, pametnim uređajima, društvenim mrežama, itd., što nije lako niti preporučljivo raditi s uparenim sustavom za upravljanje web sadržajem. Web stranica izgrađena na takvoj arhitekturi iznimno je ovisna o mogućnostima sustava koji većinski nije optimiziran za korištenje na drugim platformama. Npr. Wordpress daje korisniku na odabir razne teme i funkcionalnosti. Kada korisnik odabere što želi, ta funkcionalnost se primjenjuje.

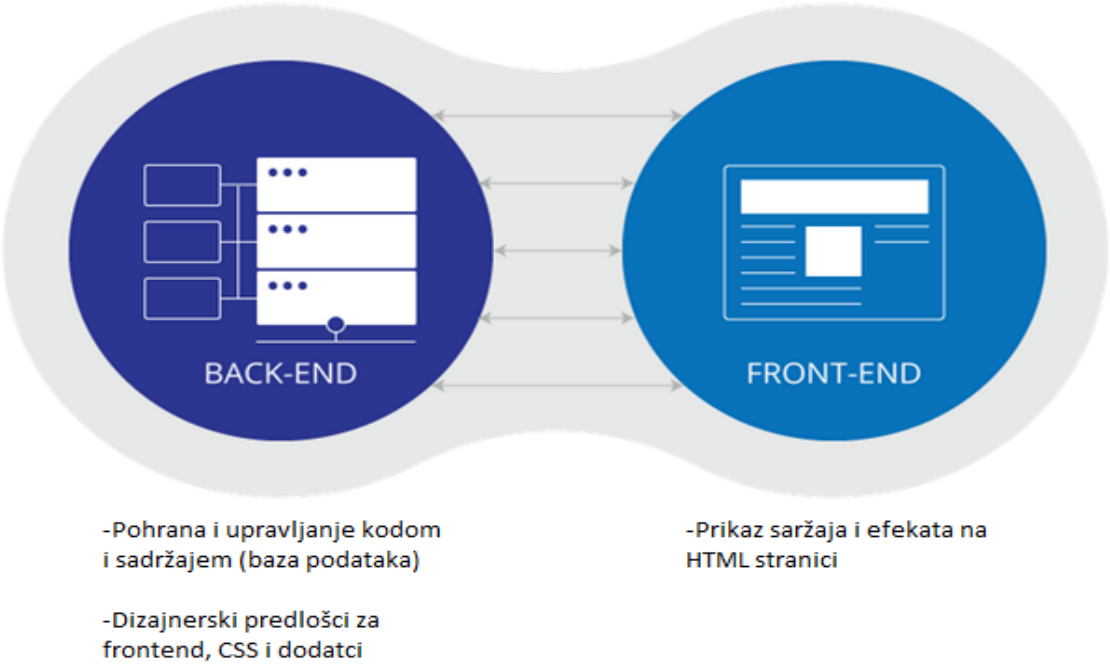
No što kada web stranica postane popularna i korisnik poželi stvoriti mobilnu aplikaciju? Treba sve graditi ispočetka ili koristiti polovična rješenja kao što je plugin za pretvorbu Wordpress stranice u mobilnu aplikaciju. Problem ovakve implementacije je u tome što Wordpress pohranjuje korisnike, detalje o temama i funkcionalnostima u bazu podataka koja je *neizostavno vezana za sustav za izradu sadržaja.* Dakle svugdje gdje je Wordpress implementiran, uz njega mora ići cijeli

sustav. Ovo stvara napuhnute i nesigurne aplikacije u kojima bismo, kada bismo željeli da se isti sadržaj istovremeno pojavi na obje implementacije, morali ručno duplo unositi sadržaj ili bitno modificirati Wordpress kako bi sadržaj povlačio iz baze A, a sve podatke za funkcionalnost iz baze B. Idealan sustav za ovakvu situaciju je raspareni sustav o kojemu će u nastavku biti riječ.

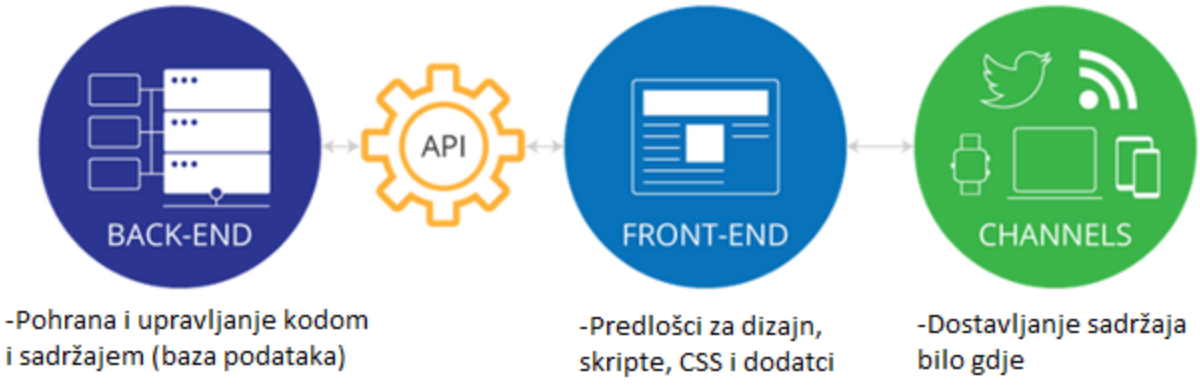
3.2. Raspareni sustavi za upravljanje web sadržajem

Kao što mu i sam naziv govori, raspareni sustav za upravljanje web sadržajem rastavlja frontend od backenda. U ovakvom sustavu API dostavlja sadržaj koji je backend procesirao frontendu, tako izolirajući backend od javne pristupnosti i čineći ga sigurnijim. Korištenje ovog sustava preporuča se za aplikacije ili za multi-kanalan web projekt. Multi-kanalnim web projektom smatraju se web stranice/aplikacije koje su dostupne na više uređaja s određenim nepromijenjenim značajkama.

Npr. Zagrebačka banka omogućava korištenje usluga poput pregleda stanja računa, uplate, itd. putem web stranice (e-zaba) ili mobilne/tablet aplikacije (m-zaba) bez gubitka sadržaja između raznih implementacija. Takav projekt zahtjeva raspareni ili bezglavi sustav za upravljanje sadržajem jer on bitno brže i efikasnije dostavlja sadržaj raznim verzijama istog projekta. Zašto je to tako?



Slika 1: Prikaz tradicionalnog CMS-a [4]



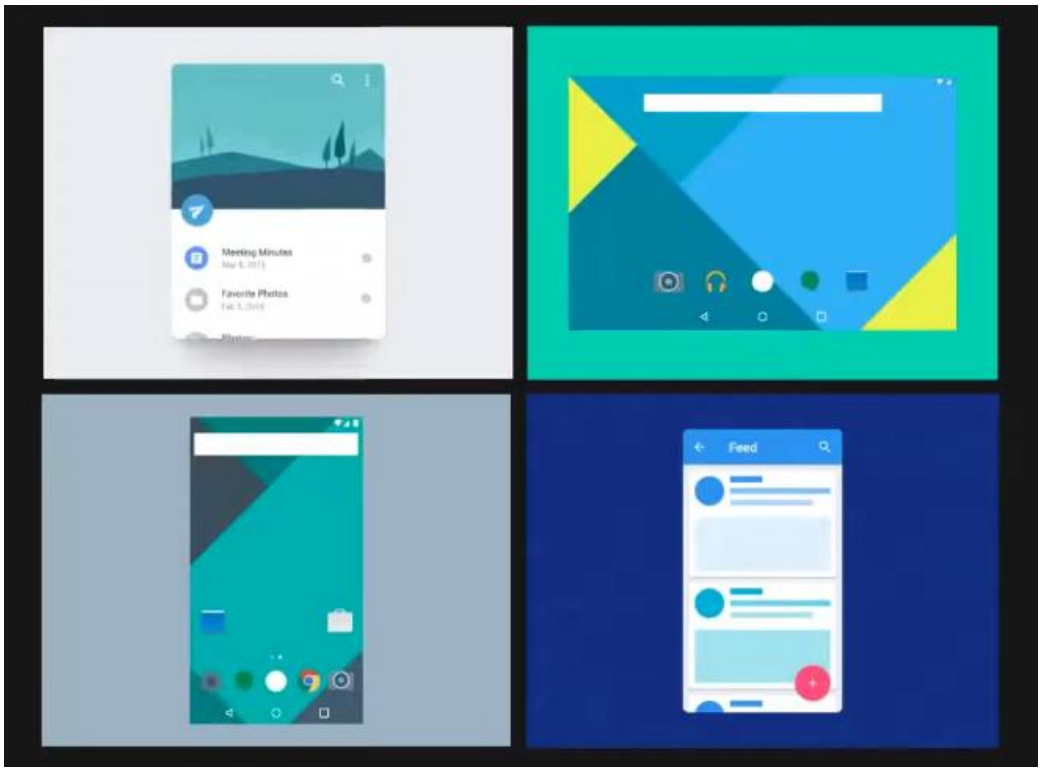
Slika 2: Raspareni sustav za upravljanje web sadržajem [5]

Kao što je vidljivo na slici 1, **tradicionalni sustav** ne razdvaja prezentaciju od funkcionalnosti. Dakle u istoj mapi pronalazimo teme, kod za funkcionalnost sustava za izradu sadržaja, CSS kod,

[4] Prilagođeno s https://www.hgsdigital.com/blogs/coupled-vs-decoupled-vs-hybrid-vs-headless-cms
[5] Prilagođeno s https://www.hgsdigital.com/blogs/coupled-vs-decoupled-vs-hybrid-vs-headless-cms

kod same web stranice itd. Ako želimo odjednom prenamijeniti taj sadržaj za mobilnu aplikaciju, neće biti moguće to kvalitetno napraviti bez građenja svega ponovno. Tu dolazimo do **rasparenog sustava**. Prema slici 2, raspareni sustav odvaja bazu podataka u kojoj je sadržaj od svih drugih funkcionalnosti, te API-em šalje sadržaj dalje.

U frontendu se obično nalaze razni moduli za dizajn (kako bi bio dosljedan na svim platformama – vidi sliku 3) od kojih dizajneri sastavljaju implementacije za razne platforme. Pošto su frontend i backend razdvojeni, dizajnerima su ruke slobodne što se tiče dizajnerskih implementacija na raznim kanalima. Dizajner može web stranicu napraviti u npr. Vue.js, aplikaciju u React.js, a implementaciju za pametni sat u Angularu, što ne bi bilo moguće s tradicionalnim sustavom. Dakle bitna razlika je u tome gdje se nalazi sustav i kako se šalje sadržaj.



Slika 3: Primjer implementacije modula za dizajn - dosljednost na svim platformama [6]

[6] Preuzeto s https://www.youtube.com/watch?v=ZVASQIKDyfU

3.2.1. Primjer

Recimo da je neka novinarska kuća implementirala ovakav sustav, platila programere i dobila gotov proizvod. Proces stvaranja članka bi tekao ovako: novinar posjeti *zasebnu domenu* na kojoj se nalazi sučelje za pisanje članaka. On napiše članak i pohrani ga. Taj članak je zatim pretvoren u notiranu datoteku (najčešće JavaScript Object Notation - JSON) koja će API-em biti poslana poslužitelju. Poslužitelj zaprima datoteku, tumači je i pohranjuje je u bazu podataka. Zatim urednik pristupa sučelju za pisanje članaka koje od poslužitelja API-em zaprima sav sadržaj. Urednik odobrava članak, te se on se API-em šalje natrag poslužitelju. On će ga od sada slati dalje na sve destinacije i domene.

3.2.2. Prednosti

1.) Sigurnost

Pošto su ovdje sustav za unos sadržaja i baza podataka pohranjeni na drugoj domeni kojoj se može ograničiti pristup i prema IP-u, ovaj sustav je gotovo apsolutno siguran od neželjenih upada sa strane frontenda i stoga bitno sigurniji od tradicionalnog.

2.) Skalabilnost

Prilikom mijenjanja ili radova na backendu sustava, na frontend se nikako ne utječe jer ne zavisi o konstantnim pozivima baze podataka, već se oslanja na aktualizaciju putem API-a. Ta funkcionalnost također omogućuje puno lakšu prilagodbu sustava većoj publici ili novoj platformi.

3.) Raznolikost

Pošto su frontend i backend odvojeni, oni ne moraju dijeliti isti programski jezik što programerima daje odriješene ruke kod konstrukcije backenda, ali i dizajnerima kod konstrukcije frontenda.

4.) **Istovremeno objavljivanje na više kanala**

Ova funkcionalnost iznimno je bitna za web projekte kojima je bitna prisutnost na svim dostupnim kanalima.

3.2.3. Nedostaci

1.) **Kompleksnost**

Ovakav sustav ne može biti instaliran bez velikog programskog znanja, niti postoji opcija jednostavne instalacije kao kod tradicionalnih sustava.

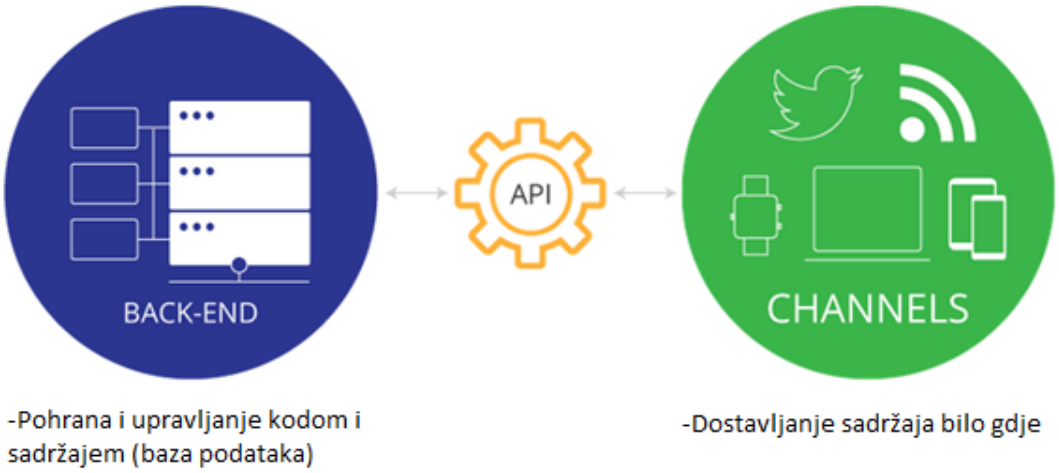
2.) Cijena

Pošto ovakvi sustavi ne mogu biti lako instalirani i zahtijevaju sposoban tim, implementacija ovakvog sustava dolazi s visokom cijenom. Na tržištu već postoje i razne besplatne opcije poput Cockpita[7], ali i za implementaciju takvog rješenja treba imati pozamašno poznavanje web programiranja i dizajnera koji će stvoriti frontend.

3.3. Bezglavi sustav za upravljanje web sadržajem

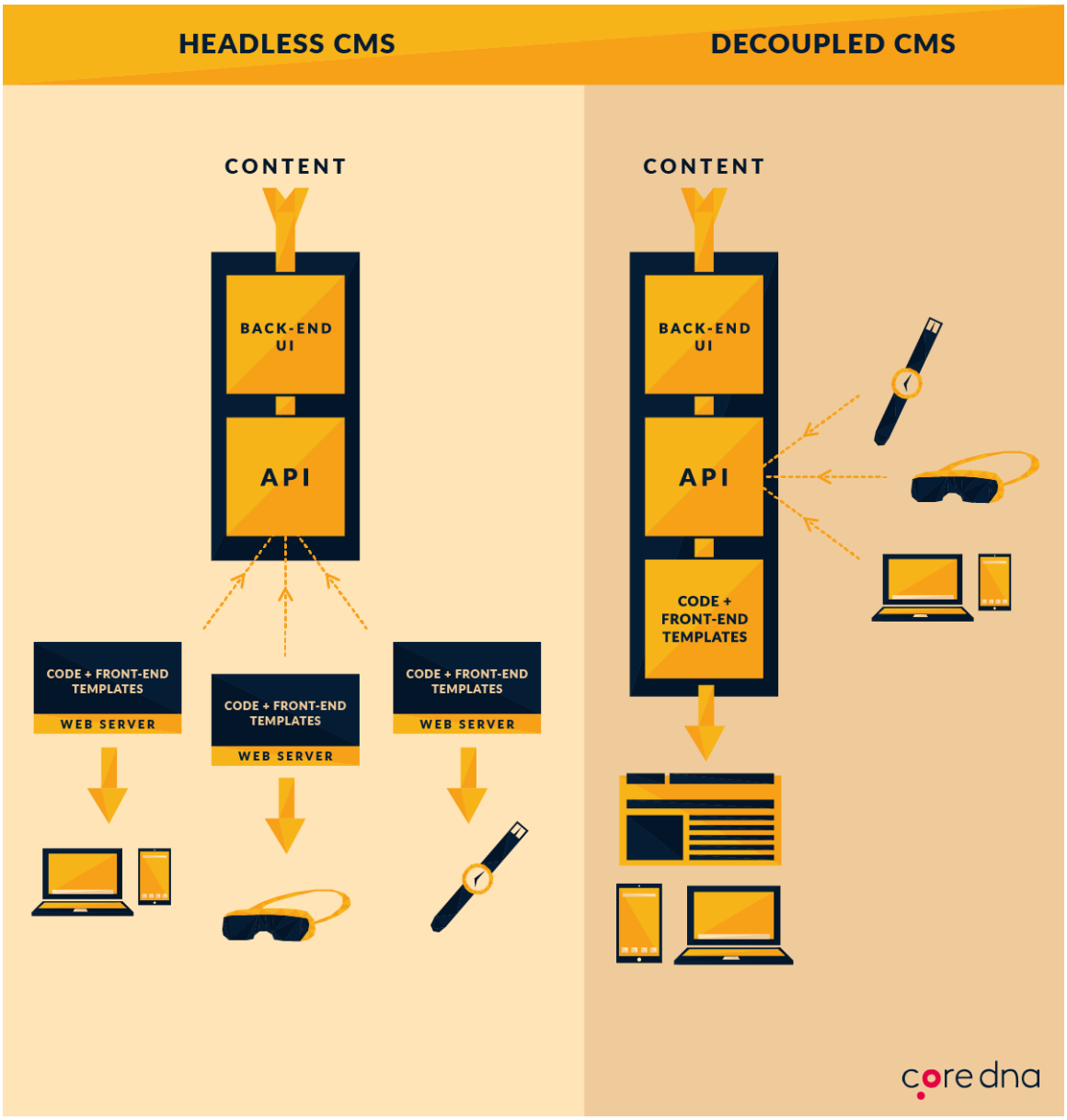
U bezglavome sustavu glavni fokus nalazi se na backendu. Ondje se pohranjuje sav sadržaj koji zatim u sirovom (engl. *raw*) formatu biva API-em poslan dalje, kao što je i vidljivo na slici 4. Po čemu se onda bezglavi sustav razlikuje od rasparenog? U tome što kod rasparenog postoji mogućnost stvaranja predložaka prema kojima se dalje gradi web stranica, dok bezglavi sustav svaku implementaciju gleda kao zasebnu jedinku (vidi sliku 5). Pošto bezglavi sustav ne podržava mogućnosti uređivanja dizajnerskih predložaka i prethodnog pregleda novog članka, ovaj sustav odgovornost za dizajn u potpunosti stavlja u ruke frontend dizajnera. Druga bitna razlika je u tome što se raspareni sustav smatra proaktivnim – on priprema sadržaj za prezentaciju i šalje ga dalje, dok je bezglavi sustav reaktivan – on pripremi sadržaj i čeka da ga netko zatraži.

[7] https://getcockpit.com/



Slika 4: Bezglavi sustav za upravljanje web sadržajem[8]

[8] Prilagođeno s https://www.hgsdigital.com/blogs/coupled-vs-decoupled-vs-hybrid-vs-headless-cms



Slika 5: Razlika između bezglavog i rasparenog sustava za izradu web sadržaja[9]

[9] Preuzeto s https://www.coredna.com/blogs/headless-vs-decoupled-cms#2

3.3.1. Prednosti

1.) **Sigurnost, skalabilnost, raznolikost i mogućnost objavljivanja na više kanala**

Pošto se bezglavi sustav za izradu sadržaja smatra podvrstom rasparenog sustava, oni dijele većinu prednosti: kao i raspareni sustav, bezglavi sustav nudi najbolju sigurnost, osiguranu skalabilnost, neograničenu kreativnost za programere i dizajnere i, finalno, mogućnost istovremenog objavljivanja na svim platformama.

3.3.2. Nedostatci

1.) Korisničke mogućnosti

U bezglavome sustavu nije moguće obaviti prethodan pregled sadržaja koji netko želi objaviti, niti je moguće koristiti dizajnerske predloške. Ovaj sustav daje najveću slobodu dizajnerima da rade što žele, ali ujedno oduzima slobodu klijenta da samostalno promijeni nešto bez plaćanja i korištenja programerskog tima.

2.) **Cijena**

Pošto predložaka nema, za optimiziranje izgleda nužna je dugoročna suradnja između klijenta i tima kako bi se postigao izgled koji klijent želi. Također, pošto prezentacijskog sloja nema, programeri ga moraju ili potpuno nanovo izgraditi ili ga skrpati od postojećih rješenja, tako čineći ovu opciju apsolutno najskupljom od svih na tržištu.

4. Wordpress

Wordpress danas koristi 63.6% korisnika sustava za upravljanje web sadržajem koje World Wide Web Consortium prati (W3Techs, 2020), te je najzastupljeniji takav sustav na tržištu. Velik broj web stranica popularnih tvrtki i ličnosti danas koristi Wordpress: Sony Playstation, Usain Bolt, Enterprise i NASA (Muriuki, 2019) samo su neka od imena koja Wordpress može uvrstiti na listu korisnika. Upravo radi toga ostatak ovoga rada pokriti će funkcionalnosti Wordpressa, njegovu povijest, prednosti i nedostatke, te ga finalno usporediti s opcijama u samostalnoj izradi.

4.1. Povijest

Povijest Wordpressa započinje još 2001. pod nazivom B2/cafelog u autorstvu Michela Valdrighia[10] (Gregersen, 2020). B2/cafelog bio je alat izgrađen na infrastrukturi PHP-a i MySQL-a korišten specifično za pisanje blogova. On je prvi implementirao dinamično generiranje web stranica iz sadržaja baze podataka, tako uvodeći novu eru stvaranja sadržaja i sustava za upravljanje web sadržajem. Valdrighi je napustio projekt 2002., no preuzeli su ga Mullenweg[11] i Little[12].

U svibnju 2003. iz B2/cafeloga osvanuo je novi projekt u novom autorstvu: Wordpress. Godinu dana kasnije Wordpressu su dodane danas nezaobilazne funkcionalnosti dodataka (plug-in), a u veljači 2005. stigle su teme i sheme (Wordpress, 2020). WordPress je 2008. po prvi puta bio i službeno priznat, te mu je dodijeljena nagrada Infoworlda „*Best of open source software awards: Collaboration*“ (Najbolje od softvera otvorenog koda: kolaboracije). Do 2011. osvojio je još cijeli niz nagrada, no ta godina ga je zacementirala kao prvaka sustava za upravljanje web sadržajem –

[10] Francuski programer koji se smatra idejnim začetnikom Wordpressa. Radi nedostatka financijske podrške, napustio je projekt b2/cafelog koji je bio prethodnik Wordpressa (Hajdarbegovic, 2019).

[11] Matt Mullenweg preuzeo je projekt b2/cafelog kada ga je njegov originalni kreator napustio i tako stvorio jedan od najpoznatijih sustava za izradu web sadržaja na svijetu – Wordpress. Mullenweg projekt je preuzeo sa samo 19 godina, te aktivno sudjeluje u njegovom održavanju i poboljšavanju od 2003. Uz Wordpress, Mullenweg danas posjeduje i tvrtku pod nazivom Automattic koja stoji iza mnoštva popularnih proizvoda kao što su Jetpack, WooCommerce, Simplenote i Longreads (Mullenweg, 2018).

[12] Mike Little pridružio se Wordpress timu u doba kada ga je Mullenweg preuzimao, ostavljajući komentar na Mullenwegovoj objavio o preuzimanju b2/cafeloga (Mullenweg, 2003). Little je za svoj doprinos dobio nagradu Konferencije pretraživanja, analitike i društvenih medija (engl. *Search, Analytics, Social Media Conference – SAScon*) za izvanredan doprinos digitalnim medijima (SAScon, 2013).

Wordpress je tada koristilo preko 50% web stranica koje su koristile javno dostupni sustav za upravljanje sadržajem, te do danas nije skinut s trona.

4.2. Značajke Wordpressa

Kao što je već spomenuto, Wordpress funkcionira kao tradicionalni sustav za upravljanje web sadržajem – sam sustav, web stranica i sučelje za objavljivanje nalaze se na istome mjestu, međusobno su isprepleteni i zavisni jedni o drugima. Cijeli Wordpress napisan je u PHP-u, a za bazu podataka koristi MySQL.

Posebna odlika Wordpressa, ali i mnogih drugih popularnih sustava za upravljanje web sadržajem, jest činjenica da ga može koristiti potpuno programerski nepismena osoba. Njegovo sučelje daje korisniku veliki opseg mogućnosti bez da od njega zahtjeva poznavanje arhitekture ili jezika sustava. Te mogućnosti su sljedeće:

1.) Dodatci (*Plug-ins*)

Dodatci zasigurno bitno pridonose broju korisnika koje Wordpress privlači – oni nude modifikacije samog sustava, dodatne opcije za SEO (*Search Engine Optimisation*), ali i još nebrojeno mnogo drugih mogućnosti.

Neki od najpoznatijih dodataka za Wordpress su dodatci koji se bave SEO-m (YoastSEO), dodatci koji se bave sigurnošću (Akismet, Wordfence), dodatci za dodavanje kontaktnih obrazaca (Contact Form 7), dodatci za e-kupovinu (WooCommerce), dodatci za Google Analytics (Google Analytics for WordPress), te dodatci za građenje stranica (Elementor).

Iz ovog popisa lako je zaključiti da praktički nema toga što Wordpress ne može, no sva ta funkcionalnost ima i svoju cijenu – napuhnuti sustav prepun dodataka čiju punu funkcionalnost rijetko tko koristi, ali svejedno učitava sve funkcije. Drugi problem s Wordpress dodatcima je sigurnost, posebno za programski nepismene. Iako većina programera svoje dodatke razvija s najboljim namjerama, mogu im se dogoditi propusti, posebno amaterima koji svoje dodatke daju besplatno.

2.) Teme

Wordpress svojim korisnicima omogućava korištenje raznih tema, odnosno dizajnerskih predložaka. Najosnovnije teme korisniku daju izbor među bojama koje želi koristiti na web stranici, te par osnovnih shema za određene stranice kao što su blog, kontaktna stranica, početna stranica itd. Naprednije teme mogu sadržavati i razne funkcionalnosti osim samog dizajna, kao npr. „*drag and drop*“ alate za izgradnju web stranica, SEO funkcionalnosti, dodatne alate za modifikaciju dizajna itd.

Iako se u Wordpressu ne može bez poznavanja određenih programskih jezika i alata izvesti svaka dizajnerska ideja, posebno neka interaktivna ili apstraktna, on nudi ogromnu bazu besplatnih i kupovnih tema u kojima će svatko zasigurno pronaći nešto što će odgovarati upravo njegovim potrebama. No, teme pate od istih problema kao i dodatci – mogu biti pretjerano napuhnute i nesigurne.

3.) Administracija

Wordpress je iznimno susretljiv što se tiče administracije – dozvoljava dodavanje beskonačnog broja korisnika kao i upravljanje njihovim ovlastima na stranici. On dijeli korisnike u 6 unaprijed definiranih skupina s unaprijed definiranim ovlastima: Super Admin, Administrator, Editor, Author, Contributor i Subscriber (Wordpress, 2020). Ukoliko je korisnik i pismen u PHP-u može samostalno dodati nove uloge i ovlasti.

4.) Sučelje za objavljivanje

U Wordpressovom sučelju lako je dodati novi članak za blog ili neku stranicu zahvaljujući njegovom sustavu kategorija. Kategorije dozvoljavaju označavanje pripadnosti određenog članka njegovoj temi, te prikazivanje istoga na odgovarajućoj stranici. Recimo da imamo web stranicu o uzgajanju bilja i glavne stranice su nam uzgoj voća, uzgoj povrća i uzgoj cvijeća. Ako određeni članak označimo pod uzgoj cvijeća, on će se (naravno ovisno i o temi), pojaviti na početnoj stranici i na stranici uzgoj cvijeća, dok će ostale zaobići.

4.3. Wordpress iza kulisa

Kada korisnik posjeti Wordpress stranicu, iza kulisa se odvija cijeli niz procesa prije nego stranica bude prikazana. Ovdje su ti procesi poredani prema redu učitavanja.

a) Učitavanje wp-config.php

Sadrži globalne varijable za Wordpress stranicu poput informacija o bazi podataka, a mogu mu biti dodane i druge funkcije poput mijenjanja zadane mape za pohranu multimedijskog sadržaja, isključivanje automatskih nadogradnji, limitiranje revizija članaka, određivanje maksimalne veličine za postavljeni sadržaj i sl.

b) Spajanje i odabir baze podataka

Wordpress se sada spaja na bazu podataka i odabire onu u kojoj je pohranjen sadržaj za web stranicu koju pokušava prikazati.

c) Učitavanje cache.php

Wordpress nudi automatsko predmemoriranje (engl. *caching*) zahtjevnih upita za bazu podataka kako isti upit ne bi morali biti ponavljani za učitavanje iste web stranice. Ovi upiti pohranjeni su samo za pristup toj stranici ukoliko programer ne doradi sustav za predmemoriranje. On može postaviti predmemoriranje na određeno vrijeme i/ili za sve stranice na određenoj domeni (Wordpress, 2020).

d) Učitavanje l10n.php

Ova datoteka učitava lokalizacijske sustave Wordpressa – prijevode i jezike.

e) Učitavanje dodataka

Wordpress aktivira sve instalirane i uključene dodatke, provjeravajući koji su uključeni u bazi.

f) Učitavanje pravila za prepisivanje

Pravila za prepisivanje nezaobilazna su za dobar SEO web stranice, pa je tako i Wordpress od verzije 4.2 (WPBeginner, 2020) kao zadano pravilo postavio SEO odobrene linkove. Prije su zadane poveznice bile bazirane na numeraciji članaka ili stranica kao npr. https://www.example.com/?p=10467, što ne govori puno o tome što čitamo, a pretraživači izgled poveznice uzimaju kao jedan od bitnijih čimbenika prilikom rangiranja rezultata. Danas Wordpress kao zadanu postavku za poveznice postavlja naziv članka ili stranice, što izgleda ovako: https://www.example.com/kategorija/naslov-mojeg-članka.

g) Instanciranje $wp_query, $wp_rewrite i $wp

$wp_query je globalna instanca u kojoj se nalazi wp_query klasa. Wp_query je verzija query funkcije koju inače koristimo za pristup bazama podataka iz php datoteke i u suštini šalje zahtjev za povratnim informacijama bazi podataka.

$wp_rewrite je globalna instanca u kojoj se nalazi wp_rewrite klasa. Unutar klase nalaze se pravila za prepisivanje koja su obrađena pod *f) Učitavanje pravila za prepisivanje.*

$wp je globalna instanca wp klase koja sadrži funkcije koje će parsirati zahtjeve.

h) Učitavanje functions.php

Functions.php je datoteka koja može sadržavati sve backend i frontend funkcionalnosti Wordpessa – dodavanje stavki na kontrolnu ploču, izmjena broja znakova dozvoljenih u članku, manipuliranje slikama unutar članka, itd. Ova datoteka direktno je povezana s temom koju korisnik koristi, te njeno učitavanja možemo protumačiti i kao učitavanje teme.

i) Određivanje korisnika

Wordpress ovdje određuje tko je korisnik i koje ovlasti ima.

j) Pokretanje wp()

Wordpress poziva wp() funkciju koja se nalazi u functions.php datoteci, te poziva $wp->main, funkciju koja postavlja sve varijable potrebne Wordpressu (Wordpress, 2020).

k) Parsiranje zahtjeva

Wordpress sada ima sve što treba kako bi parsirao korisnikov zahtjev. On provjerava pravila za prepisivanje i šalje zahtjev zaglavlja (header request).

l) Pokretanje upita

Wordpress zaprima upit i prosljeđuje ga bazi. Ova faza završava vračanjem rezultata upita.

m) Pozivanje sheme

Wordpress sada prema unaprijed zadanim kriterijima i zahtjevu korisnika sastavlja shemu web stranice koja će biti prikazana. Npr. korisnik zatraži stranicu https://example.com/kategorija/cvijece. Wordpress traži datoteku category-cvijece.php, ako je ne nađe, traži dalje prema već uhodanoj hijerarhiji finalno prikazujući index.php (Wordpress, 2020). Unutar datoteke category-cvijece.php on pronalazi naredbe za prikaz stranice kao što je učitavanje headera, učitavanje footera i raspored elemenata na stranici.

n) Gašenje

Wordpress se sada gasi i dolazi red na poslužitelj da pošalje HTML, CSS i Javascript pregledniku koji ga finalno prikazuje korisniku.

5. Wordpress ili sustav u samostalnoj izradi?

Sigurna je pretpostavka da je gotovo svaki web programer u jednome trenutku pomislio da bi trebao napraviti vlastiti sustav za upravljanje web sadržajem – netko mu je preko Wordpressa hakirao poslužitelj, klijent je instalirao 20 dodataka za nešto što je lako rješivo na jednostavniji način i stranici treba punih 5 minuta da se učita, nova verzija Wordpressa u potpunosti mu je uništila dosadašnji kod i sada mora sve prepisivati prema novim standardima, nešto je zaštekalo i on nikako ne može zaključiti što, jer ne zna svaku funkciju i interakciju Wordpressa, klijent želi mobilnu aplikaciju…

Lista razloga zašto bi netko pomislio na izradu vlastitog sustava poveća je, no je li to uistinu nužno u svakoj situaciji? Štoviše, je li ikada nužno ili samo preporučljivo?

5.1. Potreba za specifičnim značajkama

Recimo da određena tvrtka koja se bavi izradom web aplikacija dobije velikog klijenta koji je jako aktivan na mnogo kanala – npr. Večernji list. Večernji list ima svoju web stranicu, posebnu verziju stranice za mobitele i tablete, fizičke novine, profile na društvenim mrežama i aplikaciju za mobilne uređaje.

Može li se za Večernji list zalijepiti Wordpress bez ikakvih modifikacija? Nikako ne. Dok Wordpress može podržati istovremeno objavljivanje na više web stranica (npr. https://example.com i https://m.example.com) i na socijalnim mrežama uz korištenje dodataka, aktualizacija aplikacije ostaje problematična. Mora li se onda za Večernji list izgraditi potpuno novi sustav specifičan za njih? Može i ne mora. Većina tvrtki koje se bavi izradom web aplikacija koriste polovična rješenja – uzmu dio već postojećeg sustava i modificiraju ga prema svojim potrebama, u ovome slučaju, odgovor na te potrebe bio bi bezglavi ili raspareni sustav, koji može uključivati i jako modificirani Wordpress.

5.2. Poznavanje sustava

Dok preinačavanje postojećeg sustava zahtjeva manje posla, nije nužno da će taj sustav moći pružiti apsolutno sve što je za određenog klijenta potrebno, niti tim koji ga izrađuje ima apsolutnu kontrolu nad svim značajkama toga sustava. Jedna od prednosti izrađivanja vlastitog sustava za izradu web sadržaja je stopostotna kontrola nad svakom stavkom sustava, od interakcija i značajki do prezentacije.

5.3. Zaobilaženje napuhanosti

Wordpress bez dodataka dolazi s pozamašnom knjižnicom funkcija, no one ne utječu previše na brzinu učitavanja stranice, niti mogu međusobno stvoriti neželjene reakcije. Do problema dolazi kada korisnik u Wordpress dodaje dodatke i teme koje dolaze s nebrojeno nepotrebnih funkcija, pretjeranom količinom stilskih opcija i drugim dodatcima koji nepotrebno napuhuju sustav i uzrokuju sporije učitavanje stranice, a ponekad i sustavne greške.

Brzina učitavanja stranice jedna je od najbitnijih stavki kvalitetnog web dizajna – prema istraživanjima koja je proveo Walmart (Bixby, 2012) nakon prelaženja na bezglavi sustav za izradu web sadržaja, konverzije, odnosno prodaje putem web dućana bitno se povećavaju smanjenjem duljine učitavanje stranice. Također, brzina učitavanja utječe i na to kako će vas najpoznatiji web preglednici rangirati u rezultatima pretrage.

Kako bi se web stranica brzo učitala, nije nužno u potpunosti odbaciti Wordpress, ali potrebno je biti u stanju samostalno napisati dodatke i teme koje će se koristiti, ili modificirati već postojeće.

5.4. Sigurnost

Wordpress u potpunosti otvara bazu podataka prema javnosti. Kada klijent piše novi članak ili korisnik otvara stranicu, Wordpress se spaja direktno na bazu podataka tako je ostavljajući ranjivijom. Bezglavi i raspareni sustavi, oba nužno većim dijelom u samostalnoj izradi, zaobilaze ovaj problem potpuno zatvarajući bazu od očiju javnosti i šaljući podatke putem API-a.

5.5. Trošak izrade i održavanja

Dok Wordpress, kao i slične implementacije poput Drupala i Joomle, koštaju 0,00 kn i dobivaju besplatno ažuriranje, opcije u samostalnoj izradi bitno su skuplje. Tvrtka prvo mora sustav izraditi, ali ga zatim i zauvijek održavati. Svaki puta kada jezik koji sačinjava arhitekturu sustava dobije novu verziju, otkrije se neki propust ili se krpa rupa u sigurnosti, to će morati popravljati i ažurirati zaposlenici firme. Opseg posla bitno se smanjuje ukoliko firma odluči koristiti već postojeći sustav i samo ga nadogradi i izmijeni.

5.6. Dokumentacija

Prilikom izrade sasvim novog sustava, programeri koji su ga izradili jedini su koji u potpunosti znaju kako funkcionira – ne mogu uzeti novog zaposlenika i očekivati da će znati koristiti njihov sustav, već ga moraju prvo educirati. Također, za takav sustav ne postoji dokumentacija, ne postoji online zajednica u kojoj tisuće ljudi koriste isti sustav i međusobno si pomažu s greškama, samostalno ste odgovorni za pronalaženje rješenja za sve greške koje se mogu pojaviti.

5.7. Odlazak klijenta

Recimo da klijent dobije jeftiniju ponudu za održavanje svoje web stranice i želi napustiti vašu firmu. Vi ste vlasnici sustava za izradu web sadržaja koji vaš klijent koristi i ne biste željeli dijeliti svoj kod s drugom tvrtkom. Moguće je klijentu naplatiti za odlazak, ali s time ga ujedno i ucjenjujete da ostane – nitko osim vašeg tima ne zna u potpunosti koristiti taj sustav, a cijena za odlazak bit će visoka.

6. Zaključak

Od samog začetka interneta pa do danas, sustavi za upravljanje web sadržajem su evoluirali usporedno s internetom. Kako se opseg mogućnosti manipulacija sadržajem povećavao, tako su se povećavale i arhitekture sustava, pa smo s vremenom dobili danas najpoznatije, i za svoje mogućnosti zaista revolucionarne, sustave poput Wordpressa, Joomle i Drupala. Takvi, već gotovi sustavi nude cijeli niz funkcionalnih dodataka i dizajnerskih pomagala koje mogu koristiti i ljudi bez ikakvog programerskog znanja.

No oni su daleko od idealnog rješenja za velike tvrtke ili tvrtke koje žele biti istovremeno prisutne na više kanala. Za takve tvrtke, odgovor mora biti sustav bar djelomično u vlastitoj izradi - ukoliko si ga mogu priuštiti. Iako skuplji, sustav u vlastitoj izradi može svoju arhitekturu prilagoditi potrebama klijenta, te ponuditi poboljšanu sigurnost i brzinu učitavanja, značajke koje su ekstremno bitne za konverzije kupaca ili zadržavanje publike.

Danas se na tržištu nudi veliki raspon raznih vrsta sustava za izradu web sadržaja. Od tradicionalnih arhitektura, do danas najmodernijih bezglavih, od besplatnih do ekstremno skupih, od jednostavnih do zaista kompleksnih – svatko može naći nešto što odgovara njegovim potrebama, no izazov se nalazi u odabiru podobnoga sustava za određeni projekt.

Dok tradicionalni sustav za izradu web sadržaja ne zahtjeva poznavanje programskih jezika, on je napuhnutiji, sporiji i nesigurniji od drugih vrsta sustava. Kao takav, pogodan je za web stranice koje ne pohranjuju osjetljive informacije i ne zaprimaju puno posjetitelja.

Bezglavi i raspareni sustavi nude veću sigurnost i brzinu, no za bilo kakve sistemske preinake zahtijevaju prisutnost programera (Öfverstedt, 2018). Ovi sustavi pogodni su za veće projekte koji pohranjuju osjetljive informacije, očekuju veći promet i mogu si priuštiti konstantnu prisutnost web programera.

7. Literatura

1. Bateman, K. (2019). Understanding Coupled, Decoupled, and Headless CMS Platforms. Preuzeto s https://dzone.com/articles/understanding-coupled-decoupled-and-headless-cms-platforms
2. Bixby, J. (2012). Performance = Conversions: A Walmart Case Study. Preuzeto s https://powerretail.com.au/multichannel/performance-walmart-case-study/
3. Carpenter, W. J. (2011). Getting Started with IBM FileNet P8 Content Manager. Packt Publishing.
4. Connolly D. (n.d.). CGI: Common Gateway Interface. Preuzeto s https://www.w3.org/CGI/
5. Garrett, J. J. (2005). Ajax: A New Approach to Web Applications. Preuzeto s https://web.archive.org/web/20150910072359/http://adaptivepath.org/ideas/ajax-new-approach-web-applications/
6. Graham, P. (2005). Web 2.0. Preuzeto s https://web.archive.org/web/20121010024704/http://www.paulgraham.com/web20.html
7. Gregersen, E. (2020). WordPress. Preuzeto s https://www.britannica.com/technology/WordPress#ref1157140
8. Gundavaram, S. (1996). CGI Programming on the World Wide Web. (1st Ed.) O'Reilly Open Books.
9. Hajdarbegovic, N. (2019). 1 Minute Crash Course In B2/Cafelog. Preuzeto s https://www.whoishostingthis.com/resources/b2-cafelog/
10. Holst, Christian. (2013). An E-Commerce Study: Guidelines For Better Navigation And Categories. Preuzeto s https://www.smashingmagazine.com/2013/11/guidelines-navigation-categories-ecommerce-study/
11. Lee J. (2002). Open Source Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP. (1st ed.). Indianapolis: Pearson Technology Group.
12. Lemay, L. i Moncur, M.G. (1996). JavaScript. (1st Ed.) Sams.net Publishing.
13. Longman, A. D. (1998). A history of HTML. Preuzeto s https://www.w3.org/People/Raggett/book4/ch02.html

14. Majano, L.F. (2019). Learn Modern ColdFusion in 100 Minutes. Samostalno izdavaštvo.
15. Microsoft. (n.d.). Microsoft Lifecycle Policy. Preuzeto s https://support.microsoft.com/en-us/lifecycle/search?alpha=Windows%20Server%202008
16. Mullenweg, M. (2003). The Blogging Software Dilemma. Preuzeto s https://ma.tt/2003/01/the-blogging-software-dilemma/
17. Mullenweg, M. (2018). About Matt Mullenweg. Preuzeto s https://ma.tt/about/
18. Muriuki, F. (2019). 55 Big Name Brands That Use WordPress (and Why). Preuzeto s https://www.wpexplorer.com/name-brands-use-wordpress/
19. Nauk, U. i Shivalingaiah D. (2008). Comparative Study of Web 1.0, Web 2.0 and Web 3.0. 6th International CALIBER 2008. Preuzeto s https://www.researchgate.net/publication/264845599_Comparative_Study_of_Web_10_Web_20_and_Web_30
20. Öfverstedt, L. (2018). Why go headless – a comperative study between traditional CMS and the emerging headless trend. Uppsala Universitet.
21. PHP. (n.d.). History of PHP. Preuzeto s https://www.php.net/manual/en/history.php.php
22. Robie, J. (n.d.). What is the Document Object Model? Preuzeto s https://www.w3.org/TR/WD-DOM/introduction.html
23. SAScon. (2013). Unsung Internet Visionary to be Honoured at SAScon. Preuzeto s https://web.archive.org/web/20140816214421/http://www.sascon.co.uk/unsung-internet-visionary-to-be-honoured-at-sascon/
24. Shannon, V. (2006). A 'more revolutionary' Web. Preuzeto s https://www.nytimes.com/2006/05/23/technology/23iht-web.html
25. The Apache Software Foundation. (n.d.). Apache httpd Tutorial: Introduction to Server Side Includes. Preuzeto s http://httpd.apache.org/docs/current/howto/ssi.html
26. W3Techs. (2020). Usage statistics of ASP.NET for websites. Preuzeto s https://w3techs.com/technologies/details/pl-aspnet
27. W3Techs. (2020). Usage statistics of content management systems. Preuzeto s https://w3techs.com/technologies/overview/content_management
28. W3Techs. (2020). Usage statistics of PHP for websites. Preuzeto s https://w3techs.com/technologies/details/pl-php

29. Weissinger, A. K. (1999). Asp in a Nutshell: A Desktop Quick Reference. O'Reilly Media.
30. Wordpress. (n.d.). Class Reference/WP Object Cache. Preuzeto s https://codex.wordpress.org/Class_Reference/WP_Object_Cache
31. Wordpress. (n.d.). Code Reference. Preuzeto s https://developer.wordpress.org/reference/classes/wp/main/
32. Wordpress. (n.d.). History. Preuzeto s https://wordpress.org/support/article/history/
33. Wordpress. (n.d.). Roles and Capabilities. Preuzeto s https://wordpress.org/support/article/roles-and-capabilities/
34. Wordpress. (n.d.). Temple Hierarchy. Preuzeto s https://developer.wordpress.org/themes/basics/template-hierarchy/
35. World Wide Web Consortium. (n.d.). Tim Berners-Lee Biography. Preuzeto s https://www.w3.org/People/Berners-Lee/
36. WPBeginner. (2016). What is a SEO Friendly URL Structure in WordPress. Preuzeto s https://www.wpbeginner.com/wp-tutorials/seo-friendly-url-structure-for-wordpress/

Sustav za upravljanje web sadržajem: gotov proizvod ili samostalna izrada?

Sažetak

Sustavi za upravljanje web sadržajem neophodna su stavka gotovo svake web stranice. Oni klijentu daju mogućnost preinake sadržaja neovisno o njegovim programerskim sposobnostima, tako smanjujući troškove njemu, a olakšavajući i smanjujući angažman kreatoru stranice.

Ovaj radi nudi povijesni pregled nastanka takvih sustava i popratnih tehnologija, nudi pregled raznih potencijalnih arhitektura sustava i obrađuje usporedbu sustava za upravljanje web sadržajem u samostalnoj izradi s već gotovim sustavima. Danas na tržištu pronalazimo veliku količinu već spremnih sustava za upravljanje web sadržajem poput Wordpressa, Drupala, Wix-a itd., no takvi sustavi nisu nužno najbolje rješenje za svakoga i svaki projekt. Ovaj rad pokriva situacije i okolnosti u kojima je bolje koristiti već spreman sustav, ali i one u kojima je bolje izraditi vlastiti.

Ključne riječi: sustav za upravljanje web sadržajem, WordPress, web stranica, backend, frontend

Content Management Systems: Out-of-the-Box Solutions or Do it Yourself Solutions?

Summary

Content Management Systems (CMS) are an essential part of nearly every website on the World Wide Web. They give clients the ability to publish, modify and design content, without requiring knowledge of a programming language, thus reducing his spending and making the programmer's job easier.

This thesis offers a historical overview of the inception of such systems and the technologies they depend on, an overview of multiple possible CMS architectures as well as a comparison of out-of-the-box and do it yourself CMS solutions.

Today, the market is full of out-of-the-box CMS solutions such as WordPress, Drupal, Wix, etc., but such systems aren't always the best solutions for everyone and every project. This thesis covers the situations in which and out-of-the-box solution is best, but also the ones in which a DIY system should be used.

Key words: Content Management System, WordPress, website, backend, frontend