Usporedba verzija CMS sustava Joomla!

Dolić, Marko

Sveučilište u Rijeci – Odjel za informatiku

Diplomski studij informatike – Informacijski i komunikacijski sustavi

Marko Dolić

Usporedba verzija CMS sustava Joomla!

Diplomski rad

Mentor: Prof. dr. sc. Nataša Hoić-Božić

Komentor: Dr. sc. Igor Jugo

Rijeka, lipanj 2019.

Zadatak diplomskog rada

Rijeka, 20.2.2017.

Zadatak za diplomski rad

Pristupnik: Marko Dolić

Naziv diplomskog rada: Usporedba verzija CMS sustava Joomla!

Naziv diplomskog rada na eng. jeziku: Comparison of Joomla! CMS versions

Sadržaj zadatka: Joomla! je sustav za upravljanje sadržajem (CMS) koji omogućuje izgradnju web sjedišta i online aplikacija. S obzirom da je riječ o softveru otvorenog koda koji je dostupan svima te ga je lako instalirati, koristiti i proširiti različitim dodacima, predstavlja jedan od najpopularnijih programa ove namjene. Joomla! 1.0 nastala je 2005. godine i od tada se sustav kontinuirano razvija i unaprjeđuje. Počevši od verzije 3.0, Joomla! podržava responzivan web dizajn. Zadatak diplomskog rada je opisati značajke i performanse glavnih verzija CMS sustava Joomla! počevši od verzije Joomla 2.5. Naglasak treba staviti na aspekte sigurnost te mogućnosti razvoja web sjedišta prema responzivnom web dizajnu.

Mentor:

Prof. dr. sc. Nataša Hoić-Božić

Voditelj za diplomske radove:

Izv. prof. dr. sc. Ana Meštrović

Komentor:

Dr. sc. Igor Jugo

Zadatak preuzet: 6.3.2017.

(potpis pristupnika)

Sažetak

U prvom poglavlju ovog rada objašnjeni su pojmovi CM-a i CMS-a, te su navedeni problemi kojima se bave, radi boljeg razumijevanja ostatka rada. Također su navedene i osnovne karakteristike sustava za upravljanje sadržajem. U nastaku rada prolazi se kroz osnovne značajke, proširenja i arhitekturu Joomla! sustava, kao uvod u detaljnu analizu strukture direktorija, baze podataka i tijeka izvođenja Joomla! 2.5 i Joomla! 3.9 verzija sustava. Na kraju rada navedene su glavne razlike između navedenih verzija sustava.

Ključne riječi

Joomla!, CMS, sustav za upravljanje sadržajem, CM, upravljanje sadržajem

Sadržaj

1. Uvod

Danas, kada se brzina stvaranja i prikupljanja podataka eksponencijalno povećava, javlja se potreba za njihovom organizacijom i klasifikacijom kako bi ih pojedinci i organizacije uz što manju potrošnju resursa koje imaju na raspolaganju na što efikasniji način koristili, a njihovom opetovanom uporabom u izvornom ili modificiranom obliku stvarali dodanu vrijednost.

Također, prisutnost na internetu nezaobilazna je stavka današnjice kako za pojedince tako i za organizaciju, te se javlja potreba za jednostavnim, brzim, ali i sigurnim rješenjima prilikom kreiranja web sjedišta i sustava za upravljanjem sadržajem koji ih pogone.

Jedno od rješenja na navedene probleme je i Joomla sustav za upravljanje sadržajem. Ovaj open source sustav baziran na PHP-u od korisnika ne zahtijeva posjedovanje računalnih vještina i može se koristiti „out-of-the-box“, odnosno nakon instalacije ne zahtijeva dodatnu konfiguraciju kako bi se koristio, dok u isto vrijeme dopušta korisniku izradu novih i modifikaciju postojećih proširenja, kao i razvoj aplikacija na temelju Joomla programskog okvira koji se nalazi unutar sustava.

2. Sustavi za upravljanje sadržajem

Upravljanje sadržajem (CM) je proces za prikupljanje, isporuku, preuzimanje i upravljanja informacija u bilo kojem formatu. Pojam se obično koristi vezano uz upravljanje životnim ciklusom digitalnog sadržaja, od njegovog nastanka do stalnog pohranjivanja ili brisanja. (Rouse, Content Management (CM), 2017)

Pojam sadržaja razlikuje se od onoga što podrazumijevamo kao podatak, te za razliku od podatka sadržaj ima kontekst i značenje, zbog toga ga možemo klasificirati kao informaciju. Bob Boiko posvećuje cijelo jedno poglavlje upravo ovoj temi u namjeri da čitatelju objasni i definira razliku između podatka, informacije i sadržaja. Kod implementacije procesa CM u organizaciju uočava da sve počinje s jednostavnom idejom uvođenja CMS-a, ali s vremenom umjesto rješenja dolazi do novih pitanja i nedoumica. Ovaj proces implementacije naposljetku obuhvaća cijelu kompaniju, a kao glavni razlog poteškoća navodi:

Vjerujem da je korijen problema u organizacijama nepoznavanje razlike između podatka i sadržaja. Ljudi žele da sadržaj bude jednostavan i izravan kao podatak. (Boiko, 2004)

Kao sadržaj u digitalnom konceptu možemo uzeti svaku komponentu neke web stranice, pa čak i programski kod računalne datoteke pisan u Java programskom jeziku (StackOverflow i GitHub). Svaka poveznica ili navigacija na stranici može se promatrati kao sadržaj, ali da ne ulazimo preduboko u ovu temu u nastavku su navedeni tipovi sadržaj i primjeri njihove upotrebe na web stranicama.

- Tekst – članak na web portalu kao i komentari čitatelji, specifikacija proizvoda, upute za korištenje programa
- Zvuk – audio zapisi govora, pjesme
- Slika – (profilne) slike na društvenim mrežama, fotografske slike u člancima, slika autora članka
- Video – izvještaji i reportaže, priče (engl. *story*) s društvene mreže

Ukratko, sadržaj je informacija kreirana kroz urednički (engl. *editorial*) proces s krajnjom namjerom da bude konzumirana od strane ljudi kroz publikaciju. (Barker, 2016)

Kada govorimo o upravljanju sadržajem prva pomisao je značenje riječi koje vežemo uz digitalni koncept kao što stoji u definiciji CM na početku ovog poglavlja, ali ono se upotrebljava već dugo kroz povijest. Ljudski rod traži rješenja za boljim upravljanjem sadržaja već od njegova nastanka, a jedan od najranijih primjera je knjižnica u Aleksandriji

gdje je sadržaj u obliku zapisa na papirusu bio pohranjen i organiziran, a knjižničari zaduženi za upravljanje sadržajem.

Iz navedenog se može zaključiti kako problem upravljanja sadržajem nije započeo nastankom World Wide Web-a već je količina stvaranja i prikupljanja sadržaja eksponencijalno narasla, a problem upravljanja sadržajem i pronalaska novih rješenja je kritičan. Razlog ovom povećanju je to što danas svaka osoba s odgovarajućom informacijskom infrastrukturom može kreirati i pristupiti sadržaju preko web stranica, odnosno zbog same upotrebe i razvitka informacijskih tehnologija.

Internet je nastao šezdesetih godina prošlog stoljeća ali u komercijalnu upotrebu ulazi tek devedesetih godina pojavom WWW i internetskih preglednika koji su uz tekst podržavali i korištenje multimedije na web stranicama.

U uporabu ulazi i HTML koji omogućava grafički prikaz stranica te postaje jedan od temeljnih jezika za njihovu izradu. U početku njegova korištenja sadržaj web stranice namijenjen posjetiteljima nije bilo moguće odvojiti od prezentacijskog dijela stranice. To je stvaralo poteškoće kod redizajniranja stranica koje je zahtijevalo mnogo posla i vremena čak i kod malih preinaka. Uzmimo za primjer slučaj u kojem želimo dodati stavku u navigaciju stranice, istu izmjenu je bilo potrebno dodati na sve postojeće stranice koje su već imale ovu navigaciju. Osim tehničkog dijela problema kreatori stranica nisu imali jasan uvid u sadržaj, nije postojala mogućnost njegova odvajanja i zasebnog spremanja. Iz ovih razloga javlja se potreba za sustavima za upravljanje sadržajem (engl. *Content management systems*).

Sustav za upravljanje sadržajem (CMS) je programski paket koji pruža neku razinu automatizacije za zadatke koji su potrebni za učinkovito upravljanje sadržajem. CMS je obično serverski zasnovan (engl. *server-based*), višekorisnički softver koji komunicira sa sadržajem u repozitoriju. Repozitorij može biti smješten na istom serveru, kao dio istog programskog paketa ili potpuno odvojen na zasebnom serveru. CMS omogućuje urednicima stvaranje novog sadržaja, uređivanje postojećeg sadržaja, obavljanje uređivačkih procesa na sadržaju i u konačnici sadržaj učiniti dostupnim za konzumaciju od strane drugih ljudi. (Barker, 2016)

2.1 Značajke i prednosti korištenja sustava

Pojavom CMS-a smanjena je potreba za pisanjem koda a u nekim slučajevima ono je i potpuno izbačeno, iz tog razloga kreiranje web stranica postalo je dostupno i korisnicima bez tehničkog znanja o onome što se odvija u pozadini stranice. Korištenjem predložaka (engl.

templates) korisnik dobiva gotov prezentacijski dio stranice koji se uz jednostavne izmjene može personalizirati, a na korisniku je da stranicu popuni sadržajem.

Sadržaj je odvojen od prezentacijskog dijela, te se s lakoćom može koristiti na više mjesta, odnosno on se ne stvara kod svake pojave na stranici već se dohvaća i prikazuje s jedne lokacije na sustavu. Zbog toga kod izmjene sadržaja učinjene na jednom mjestu on se ažurira kod svake pojave tog sadržaja na cijelom web sjedištu.

Također zbog odvojenosti sadržaja od prezentacije moguće je sadržaj objavljivati u različitim formatima, tako se koristeći tekst stranice može kreirati PDF datoteka istog sadržaja ili ga objaviti putem drugih kanala. Primjer objave na drugom kanalu bila bi objava naslova teksta i linka na članak putem socijalnih mreža. Znači da sadržaj može biti objavljen mnogo puta i u različitim prezentacijskim formama.

Koristeći agregaciju CMS omogućava korisniku grupiranje sadržaja radi bolje organizacije ili prezentacije sadržaja. Neki od primjera agregacije su navigacija na web stranicama i prikaz rezultata pretraživanja kada se korisniku grupira sadržaj na temelju upita.

Kako bi se uspješno odvoji prezentacijski dio od sadržaja i agregacija dala dobre rezultate sadržaj mora biti pravilno strukturiran, odnosno prije implementacije sustava trebalo bi definirati model sadržaja o kome je više rečeno u odlomku Koraci prije uvođenja sustava.

Kreiranje forma, posebno jednostavnih oblika, u CMS-u je izvedeno na način da ih i osobe bez tehničkog znanja mogu s lakoćom izraditi. Jedna od takvih je forma za unosa podataka od strane korisnika radi komunikacije s vlasnikom stranice, odnosno kontakt forma koja se nalazi na većini web sjedišta. Nakon ispunjavanja forme podaci se u većini slučajeva spremaju u CMS ili prosljeđuju na određenu e-mail adresu.

Uvedena je kontrola sadržaja te je kod sustava upravljanja sadržajem predstavljena funkcionalnost kreiranja verzija sadržaja, a time i ponovno korištenje njegovih starijih verzija. Također je smanjena mogućnost gubitka sadržaja njegovim brisanjem a utjecaj zlonamjernih radnji sveden je na minimum jer svaka verzija sadrži informaciju o korisniku koji je napravio izmjenu a promjene se lako vraćaju na prijašnje stanje.

Suradnja među korisnicima i praćenje statusa u kojem se sadržaj nalazi postignuta je upotrebom tijeka rada (engl. *editorial workflow*), tako svaki od sudionika zna u kojem se stadiju uređivačkog procesa sadržaj trenutno nalazi i za koji je stadij tko i kada zadužen.

Uvedene su još neke funkcionalnosti kako bi se suradnja između korisnika dodatno unaprijedila, tako neki sustavi nude mogućnost kreiranja i dodjeljivanja zadataka, što je povezano s tijekom rada ali i ne mora biti, diskusije i komentari vezani uz neki zadatak ili sadržaj, te mogućnost komunikacije putem chat-a.

Korištenjem dozvola svakom se pojedincu ili skupini pojedinaca (odnosno grupi) dodjeljuje uloga (engl. *role*). Pomoću njih korisnicima se može ograničiti pristup pojedinim radnjama, alatima i dijelovima sustava kako bi se smanjila opasnost zlonamjernih ili slučajnih radnji za koje korisnik nije ni svjestan a mogu prouzročiti veće promjene na sustavu.

Iako skoro svaki CMS ima različiti naziv za grupe možemo ih podijeliti na:

- Super admin – dodijeljene su sve dozvole
- Administrator – za razliku od super admina može mu biti zabranjen pristup nekim back-end dijelovima sustava ili mogućnost kreiranja i brisanja web stranica/sjedišta
- Autor/urednik/izdavač – kreira, mijenja i objavljuje sadržaj
- Gost – registrirani korisnik
- Anonimni gost – neregistrirani korisnik, za razliku od gosta može mu biti zabranjeno objavljivanje komentara

Navedene grupe su zadane odmah po instalaciji sustava a korisniku je dana sloboda da kreira ili mijenja nazive grupa i njihova prava pristupa.

Još neke od značajki sustava su personalizacija gdje se prikaz sadržaja prilagođava određenom korisniku ili grupi korisnika. Analitika koja daje informacije o ponašanju korisnika na web stranici te tako omogućava da se određenog korisnika smjesti u grupu te zatim primjeni personalizacija sadržaja. Pomoću analitike je moguće pratiti i rezultate prodaje ili broja pregleda nekog proizvoda. Korištenje višejezičnosti u prikazu sadržaja i sučelja CMS-a također je podržano u većini sustava.

Sve navedene značajke štede vrijeme i time povećavaju efikasnost urednika. Pružajući alate i funkcionalnosti dopuštaju urednicima da stvaraju više sadržaja u kraćem vremenskom roku i s manje napora. Također povećavaju i samu sigurnost sustava i smanjuju opasnost od gubitka kreiranog sadržaja.

2.2 Podjela sustava

Sustave za upravljanje sadržajem moguće je podijeliti u skupine, i to po tipu sadržaja kojim rukuju. Uobičajena podjela CMS-a, Barker ih još naziva i velikom četvorkom, je:

- Web content management (WCM) – sustav za upravljanje web sadržaja, pogodan je za pojedince i organizacije koje većinu sadržaja objavljuju na internetu. Ima ugrađene alate koji korisnicima omogućuju kreiranje, objavljivanje i kontrolu nad sadržajem web stranica
- Enterprise content management (ECM) – sustav za upravljanje sadržajem i podacima povezanim s organizacijskim procesima kao što su e-mail poruke i dokumenti. Koriste se u organizacijama kako bi se konsolidirao sadržaj koji one posjeduju i povećala produktivnost
- Digital asset management (DAM) – posebnost sustav za upravljanje digitalnim sadržajem je to što on nije namijenjen sadržaju u tekstualnom obliku već grafičkom i multimedijskom sadržaju i sadrži alate za njegovu izradu i transformaciju
- Records management (RM) – sustav za upravljanje sadržajem nastalim tijekom transakcija i zapisi stvoreni tijekom odvijanja poslovnog procesa, kao naprimjer ugovor ili transakcija nastala prodajom

Osim navedenih primjera postoji još neki specijalizirani sustavi, kao naprimjer sustavi za upravljanje dokumentima (engl. *Document management systems - DMS*) ali nisu navedeni zato što su razlike između sustava jako male a neki od gore navedenih već sadrže funkcionalnosti ovih sustava, ili barem dio njih. WCM i ECM sustavi većinom imaju mogućnost izvođenja uređivačkih procesa i objavljivanja dokumenata i multimedijskog sadržaja, što znači da integracijom s drugim sustavima pružaju i njihove funkcionalnosti, u ovom slučaju pomoću integracije s DMS-om i DAM-om. Te funkcionalnosti mogu biti ugrađene u core sustava ili dostupne za korištenje putem proširenja (engl. *extensions*).

Ova raznolikost sustava je nastala iz potrebe da se organizacijama i pojedincima omogući da izaberu proizvod koji najbolje odgovara njihovim potrebama ili standardima industrije kojom se bave.

2.3 Pregled sustava

Svaki softver podijeljen je na front-end i back-end, gdje prvi predstavlja prezentacijski dio aplikacije koji korisnik vidi i s kojim vrši interakciju, dok drugi sadrži poslovnu logiku i sloj zadužen za pristup podacima. Kod CMS-a ovi pojmovi imaju još jedno značenje, tako se front-end smatra javno vidljiva web stranica koja se prikazuje krajnjem korisniku a back-end je administrativna aplikacija koju administratori i urednici koriste za podešavanje postavki

sustava i upravljanje sadržajem. Ali naravno, bez obzira na ovu podjelu, web stranica i administrativni dio također imaju front-end i back-end.

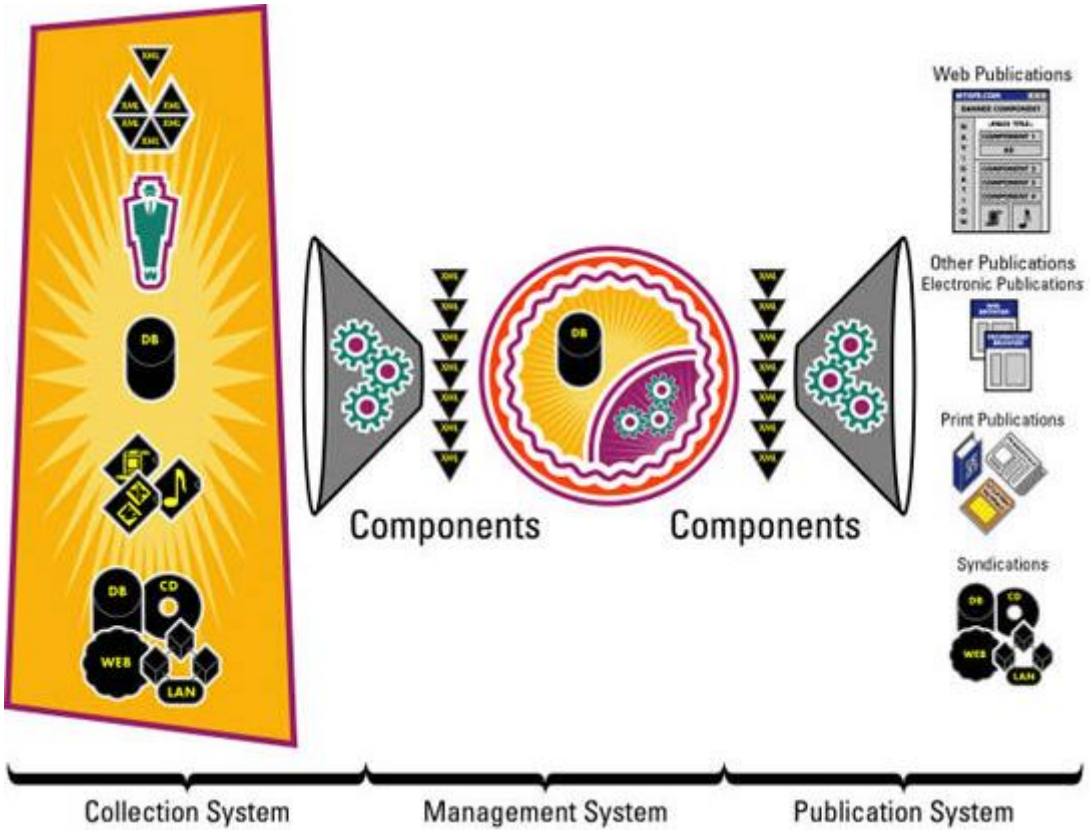
Komponente administrativne aplikacije CMS-a dijele se na aplikaciju za upravljanjem sadržajem (engl. *content management application* ili CMA) i aplikaciju za isporuku sadržaja (engl. *content delivery application* ili CDA)

CMA je front-end komponenta CMS-a (odnosno administrativne aplikacije). CMA sučelje omogućava korisniku da kreira i upravlja korporativnim ili web sadržajem. CMA obično sadrži predloške koji automatiziraju mnoge repetitivne aspekte kreiranja i editiranja sadržaja. Većina CMA-a pruža WYSIWYG[1] sučelje, koje omogućuje korisnicima kreiranje i rad sa sadržajem bez potrebe za korištenjem HTML-a. (Rouse, Content management application (CMA), 2011)

CDA je back-end komponenta administrativne aplikacije. CDA je u osnovi komponenta koja rukuje svim operacijama u pozadini dok obavlja transakciju podataka između repozitorija sadržaja i pregleda tako da kompajlira korisnikov unos kroz CMA. (Rouse, Content management system (CMS), 2016)

Bob Boiko u knjizi Content Management Bible prikazuje CMS kao cjelinu koji se sastoji od tri međusobno povezna sustava (slika 1), gdje se od podataka i informacija u sustavu za prikupljanje kreira sadržaj koji se sprema u bazu, odnosno sustav za upravljanje, te na kraju pomoću sustava za objavljivanje pretvara u publikaciju.

[1] Što vidiš to i dobiješ (engl. *what you see is what you get*) sučelja omogućavaju korisnicima da tijekom stvaranja sadržaja vide kako će on izgledati kad se objavi



Slika 1 shematski prikaz CMS-a (Boiko, 2004)

Također navodi da se granice između sustava preklapaju pa sustav upravljanja može biti dio sustava prikupljanja jer se ponekad sadržaj vraća na doradu nakon spremanja, dok sustav upravljanja smatramo dijelom sustava za objavljivanje kada se on nalazi uz Web stranicu koja ga koristi.

Sustav za prikupljanje sadrži alate za kreiranje, prikupljanje, konverziju i agregaciju sadržaja. Odnosno zadužen je za sve procese koji se izvode nad sadržajem prije trajnog spremanja ili publikacije.

Sustav za upravljanje sadrži bazu podataka u kojem je sadržaj spremljen (repozitorij), administraciju u kojoj se postavlja i mijenja konfiguracija CMS-a, te workflow.

Sustav za objavljivanje, kao što i sam naziv kaže, namijenjen je kreiranju publikacije tako da se predloške puni sadržajem i zatim dostavlja krajnjem korisniku na korištenje.

2.4 Koraci prije uvođenja sustava

Primjena CMS-a oslobađa mnogo ljudskih resursa potrebnih za procese vezane uz upravljanje sadržajem i razmjenu informacija unutar pojedine skupine ljudi. Ali bez obzira na sve

prednosti, uvođenje sustava zahtjeva velike količine resursa posebno kod korporacija, a pred korisnika stavlja mnoga pitanja od kojih su neka:

- Koji CMS će biti implementiran?
- Koja tehnologija će biti korištena u CMS-u?
- Koji sadržaj će biti spremljen i kreiran u CMS-u?
- Kako će sadržaj biti strukturiran i u koje kategorije će biti podijeljen?

Kako bi implementacija sustava prošla što uspješnije potrebno je odraditi određene korake prije samog uvođenja CMS-a.

Analiza organizacije je početni korak gdje je glavni cilj upoznati sadržaj kojim ona upravlja kao i samu strukturu organizacije i poslovne procese unutar nje. Važno je znati što i kako organizacija radi, koji su planovi organizacije i tko je zadužen za obavljanje pojedinog posla unutar organizacije. Koja vrsta publikacije se kreira i koliko je ona trenutno prilagođena za uvođenje u CMS.

Osim publikacije mora se obratiti pažnja i na sami sadržaj i postaviti pitanja vezana uz njegovu vrstu, količinu i strukturu. Da li organizacija ima razvijen pristup korisnicima i tko su oni. Koje tehnologije će biti korištene u CMS-u i koje značajke (engl. *features*) će biti ugrađene u CMS, te jesu li ovi zahtjevi dobro formirani ili su samo navedeni kao mišljenja, odnosno želje korisnika.

Na temelju izvršene analize postavljaju se ciljevi i plan njihova ostvarivanja, te se definiraju zahtjevi korisnika.

Nakon analize slijedi izrada tehničkog plana gdje je potrebno razjasniti sva pitanja vezana uz vizualne elemente CMS-a kao i one koji se nalaze u pozadini a nisu direktno vezana uz izgled. Također u ovom koraku je uključena i izrada plana sadržaja u kojoj je potrebno definirati strukturu, kategorizaciju i plan objavljivanja sadržaja.

Kod vizualnog izgleda stranice polazi se od izrade nacrta stranica (engl. *wireframes*) i mape (engl. *sitemap*) web sjedišta, te je za svaku skicu potrebno postaviti pitanja da bi se otklonile nejasnoće prije same izrade o tome kako element funkcionira, da li je dio veće cjeline, da li se pojavljuju samo na jednom ili više mjesta, koliko se često sadržaj mijenja, te kako bi se eliminirali elementi koje nije moguće implementirati.

Kod pregleda pozadinskih procesa postavljaju se pitanja o tome kako se odvija tijek uređivačkog rada, da li autor kreira sadržaj direktno u CMS-u i da li su potrebna odobrenja prije objave, koja dopuštenja postoje, da li je sadržaj lokaliziran i na kojim jezicima, da li se koriste verzije i za koji tip sadržaja, da li sadržaj već postoji i ako postoji hoće li ga biti potrebno migrirati.

Nakon što se odgovori na ova pitanja slijedi izrada plana sadržaja u kojoj se analizira struktura i agregacija sadržaja. Također se određuju grupe korisnika i dodjeljuju im se dopuštenja, izrađuje se strategija migracije, radi se pregled svih vanjskih sustava s kojim će CMS biti povezan i na koji način će povezivanje biti postignuto. U ovom koraku korisno je napraviti plan održavanja sustava koji se sastoji od plana objavljivanja sadržaja i predviđanja mogućih promjena u poslovanju, a samim time i nadogradnji sustava.

Kako bi CMS znao na koji način spremati, upravljati, pretraživati i pružiti uređivačke alate za određeni sadržaj potrebno je prethodno napraviti model sadržaja (engl. *content model*) koji je definiran tipovima, atributima i relacijama između sadržaja.

Tip sadržaja je logična podjela sadržaja po strukturi i svrsi, te se sastoji od atributa. Primjer tipa je novinski članak koji se može sastojati od sljedećih atributa:

- Kategorija
- Naslov
- Kratki opis
- Autor
- Vrijeme objave
- Tekst članka
- Tagovi

Relacije se kod CMS-a koriste s istom svrhom kao i kod relacijskih baza podataka, odnosno kako bi se stvorile veze između sadržaja. Tako u prije navedenom primjeru novinskog članka atribut kategorija definira relaciju između članka i stranice s člancima te kategorije.

Na kraj, u slučaju da je CMS koji će se koristiti već odabran radi se pregled svih funkcionalnosti koje dolaze sa sustavom i raspravlja se o prilagodbama koje je potrebno napraviti, u suprotnom slijedi odabir CMS-a.

Kod analize funkcionalnosti u obzir se mora uzeti budžet dostupan za implementaciju i koliki će se dio budžeta, ako je razvoj nove funkcionalnosti potreban, izdvojiti za istu. Također treba

sagledati tehničko znanje urednika, poznavanje HTML-a i CSS-a, i razmotriti količinu strukturnih promjena koje će korisnici moći napraviti bez posredstva developera.

2.5 Odabir sustava

Današnje tržište pruža široki spektar rješenja za upravljanje sadržajem, a na korisniku je da odabere ono koje najbolje odgovara njegovim potrebama. U idealnom slučaju ovaj odabir temelji se na provedenoj analizi i izrađenom tehničkom planu.

Kod odabira sustava polazi se od toga koje zadaće CMS treba obavljati, odnosno da li je sustav specijaliziran za rješavanje problema koji je korisniku potreban. Pojam web stranice je širok i pokriva veliki broj stranica koje se u svojoj namjeni razlikuju. Sustav koji je specijaliziran za objavljivanje vijesti gdje se svakodnevno objavljuju članci neće najbolje poslužiti kompaniji koja se bavi internetskom trgovinom zato što je arhitektura koja se nalazi u pozadini većinom predviđena za rješavanje jednog problema.

Bez obzira na gore navedeno postoji tendencija u razvoju sustava koja teži stvaranju sustava koji pokriva sve probleme vezane uz upravljanje sadržajem, ali time dolazi do razvoja kompleksnih sustava s mnoštvom značajki koje sustav mogu učiniti nepreglednim te time neprivlačnim krajnjem korisniku. Također sustavi postaju sve više generički i u namjeri da budu namijenjeni za sve često se dogodi da ne ispuni nijednu zadaću dovoljno dobro.

U slučaju da organizacija već koristi određene tehnologije i ne želi ih iz određenog razloga mijenjati ili su preduvjeti pri odabiru takvi da se zahtjeva upotreba određenih tehnologija treba uzeti u obzir i ovaj aspekt CMS-a. Primjer toga bila bi kompanije koja ima zaposlene programere za određeni programski jezik ili okvir (engl. *framework*) te je time limitirana pri razvoju i modifikaciji CMS-a.

Pod pojmom tehnologije (engl. *technology stack*) misli se na skup jezika ili okvira, bazi podataka i potpornog software na temelju kojeg aplikacija radi. Kod CMS-a back-end tehnologije su programski okviri, programski jezici, serveri (web serveri i baze podataka), operacijski sustavi i sami CMS, dok su front-end tehnologije sačinjene od programskih jezika i okvira namijenjenih izradi vizualnih sučelja, kao i jezici za označavanje (engl. *markup languages*) te stilski jezici.

Iako je tehnologija korištena kod CMS-a raznovrsna, najkorištenije su sljedeće:

- LAMP/WAMP – skraćenica za Linux/Windows operativni sustav, Apache server, MySQL relacijsku bazu podataka, i jedan od sljedećih programskih jezika Perl/PHP/Python
- ASP.NET
- Java
- Ruby on Rails
- Python (Django programski okvir)

Korisnik naravno ima mogućnost izrade vlastitog CMS-a, ali osim toga može birati između open-source i komercijalnog software.

Open-source sustavi za upravljanje sadržajem su većinom besplatni i široko dostupni korisnicima. Ovi sustavi imaju zajednicu sačinjenu od programera i dizajnera koji pridonose projektu tako da razvijaju sustav, stvaraju priključke (engl. *plugin*) i predloške. Neki priključci i predlošci su besplatni dok se drugi naplaćuju.

Komercijalni sustavi su razvijeni od strane jedne kompanije i obično zahtijevaju kupnju licence od strane korisnika. Dolaze s ugrađenim funkcionalnostima i priključcima te ne podržavaju korištenje istih razvijenih od treće strane.

U nastavku je dana tablica s prikazom glavnih razlika između dva sustava:

Tablica 1 Razlike između open-source i komercijalnih sustava

Open-source sustavi	Komercijalni sustavi
• Korištenje se ne naplaćuje • Kod većine sustava široki izbor pri odabiru agencija/developera za razvoj i prilagodbu sustava • Veliki broj priključaka, dodataka i predložaka razvijenih od strane zajednice • Veći rizik od napada na sustav zbog otvorenog koda i velikog broja korisnika	• Web hosting, održavanje stranice, podrška i nadogradnje uključene uz sustav • Konzistentno korisničko sučelje • Ne dopušta korištenje priključaka razvijenih od treće strane • Veća sigurnost sustava • Posebno osposobljen tim za podršku korisnicima • Uključeno početno osposobljavanje korisnika za rad na sustavu

Ovisno o mjestu hostinga sustavi mogu biti smješteni lokalno u data centru kompanije ili kod odabranog pružatelja hosting usluga. U slučaju da je odabran potonji kompanija može imati kontrolu kreiranjem računa kod pružatelja usluga ili kontrolu u potpunosti prepustiti pružatelju, što je slučaj kod korištenja komercijalnog CMS-a gdje je upravljanje često uključeno u paketu pri kupnji sustava.

Kao i kod ostalih aplikacija tendencija je da se lokalni hosting zamjenjuje rješenjima u oblaku (engl. *cloud*). Tako se sve više sustava za upravljanje sadržajem na tržištu nudi kao SaaS (engl. *Software-as-a-Service*) rješenje. Ovime se korisnik preplaćuje na korištenje sustav koji je smješten kod pružatelja usluga te izbjegava potrebu za instalacijom i konfiguracijom sustava.

Kod odabira sustava i koracima koji se poduzimaju prije implementacije sustava poželjno je da su u proces uključeni korisnici sustava, administratori, razvojni tim i menadžment u slučaju da se radi o uvođenju sustava u kompaniju.

3. Joomla! sustav za upravljanje sadržajem

Joomla! je besplatan i open-source sustav za upravljanje web sadržajem (WCM) koji se zasniva na Joomla web aplikacijskom programskom okviru. Ovaj okvir može se koristiti i samostalno za izradu web aplikacija a bazira se na model-view-controller (MVC) softverskom arhitektonskom uzorku.

Sustav je moguće instalirati lokalno na računalu korisnika što zahtjeva instalaciju dodatnih aplikacija i podešavanje konfiguracije kako bi sustav ispravno funkcionirao. Također, Joomla preko određenih pružatelja usluga nudi mogućnost korištenja sustava kao SaaS rješenje, pa tako pružatelj usluga preuzima na sebe posao kreiranja i ažuriranja sustava.

Naravno pružatelj naplaćuje naknadu tako da korisnik odabere plan pretplate koji mu najviše odgovara te plaća pretplatu pružatelju u određenom vremenskom intervalu, ovisno o izabranom planu. Ovim rješenjem korisnik dobiva uslugu bez da brine o stanju sustava, a nakon odabira plana pretplate kreira korisnički račun i domenu kod pružatelja.

3.1 Povijest razvoja

Sustav za upravljanje sadržajem Mambo razvijen je 2000. godine, a godinu nakon izlaska izdan je pod GNU General Public licencom te ubrzo skuplja veliku podršku zajednice. Radi spora oko autorskih prava 2005. godine većina članova Mambo Core tima daje ostavku i osniva neprofitnu organizaciju pod nazivom Open Source Matters. Iste godine izlazi prva službena verzija Joomla 1.0 CMS-a koja se nije pretjerano razlikovala od svojeg prethodnika, te je pri izlasku u sustavu bilo ispravljeno nekoliko grešaka i sigurnosnih propusta, ali u tom trenutku razvoj ovih dvaju sustava kreće u različitim smjerovima.

Kako je podrška za svaku verziju Joomle vremenski ograničena odlučeno je da će svaka x.5 verzija biti podržana kao LTS[2] dok će ostale verzije biti podržane kao STS[3] verzije.

U sljedećoj verziji Joomla 1.5, koja izlazi 2009. godine, sustav dobiva velike promjene u gotovo svim segmentima. Programski okvir se bazira na MVC-u i primjenjuje se objektno-orijentirana paradigma pri izradi sustava. Korisničko sučelje temeljito je prerađeno kako bi se poboljšala upotrebljivost i pojednostavilo korištenje, a u uporabu ulazi novi API (engl. *Application programming interface*) koji pojednostavljuje izradu ekstenzija sustava. Također

[2] LTS (engl. *long term support*) je pojam koji označava da će verzija biti podržana 18 mjeseci od izlaska, odnosno 3 mjeseca nakon izlaska sljedeće LTS verzije

[3] STS (engl. *short term support*) je pojam koji označava da će verzija biti podržana 1 mjeseca nakon izlaska sljedeće STS ili LTS verzije (verzije 1.6 i 1.7)

se proširuje podržani skup znakova, uvode jezici koji se pišu s desna na lijevo, te omogućava prijevod statičkog teksta u više jezika na front end-u i administracijskoj strani sustava, odnosno uvodi se internacionalizacija. Što se tiče korištenja predložaka dizajnerima se olakšava njihovo korištenje i daje veća kontrola pomoću novog sučelja za konfiguraciju predložaka.

Treća verzija Joomle izlazi 2012. godine pod verzijom Joomla 2.5 koja donosi 26 novih značajki i novi API, te se time dodatno olakšava korištenje sustava. Također uvodi se i mogućnost ažuriranja jednim klikom, a sustav prvi puta osim MySQL-a podržava i druge baze podataka.

Joomla 3.x serija se u prvim verzijama prvenstveno orijentirala na prilagodbe vezane uz mobilne uređaje. Kako bi se olakšalo korištenje i unaprijedilo korisničko iskustvo prilagođava se front end i administratorska aplikacija sustava.

U ovoj seriji, u odnosu na Joomlu 2.5, cijeli je sustav prošao kroz promjenu. Neke od promjena i poboljšanja vidljiva su kod uređivača sadržaja, višejezične podrške, sigurnosti sustava, izradi i pretraživanju proširenja, izradi predložaka i formi, te novi API.

Uvode se i neke nove značajke kao korištenje tagova, microdate[4], isticanje promjena između verzija sadržaja, nadogradnja jednim klikom[5].

Kako bi sustav pratio promjene u zakonima i regulativama, vezano prvenstveno uz stupanje na snagu GDPR-a, u Joomli 3.9 predstavljeno je proširenje pod nazivom Privacy Tool Suite.

Počevši od Joomle 3.3.1 donijeta je nova strategija razvoja sustava i odlučeno je da će se koristiti Semantičko verzioniranje[6] kod numeracije verzija, te se tom odlukom odustaje od ranije korištenog LTS i STS označavanja.

U skladu sa semantičkim verzioniranjem Joomla koristi sljedeći format pri numeracije verzija s ciljem da korisnik lako prepozna stupanj promjene između verzija:

[major].[minor].[patch]

[4] Specifikacija (HTML atributa) za postavljanje meta podataka unutar sadržaja na web stranicama kako bi tražilice, pretraživači i preglednici razumjeli informacije koje se nalaze na stranici te time ponudili relevantnije rezultate korisniku

[5] U ranijim verzijama trebalo je napraviti migraciju sustava između glavnih verzija, dok se za Joomlu 2.5 radi mala migracija, odnosno proširenja uključena u jezgru Joomla sustava se nadograđuju jednim klikom dok proširenja trećih strana treba nadograditi ručno

[6] Skup pravila i zahtjeva koji određuju dodjeljivanje brojeva verziji i načina kada se povećavaju. Broj verzije i način na koji se mijenja prenosi značenje osnovnog koda i onoga što je modificirano između verzija. (Preston-Werner, n.d.)

gdje je:

- Povećanje glavne (engl. *major*) verzije označava promjenu koje prekida kompatibilnost između verzija
- Povećanje manje (engl. *minor*) verzije označava značajniju promjenu postojećih ili dodavanje novih značajki sustavu
- Zakrpa (engl. *patch*) verzije označava ispravak grešaka i sigurnosnih propusta sustava

Uvođenjem nove strategije promijenjeno je trajanje podrške sustava, odnosno glavnoj verziji može biti proglašen kraj životnog vijeka koji nastupa 2 godine od izlaska zadnje manje verzije sustava dok se podrška manjoj verziji ukida izlaskom sljedeće manje verzije.

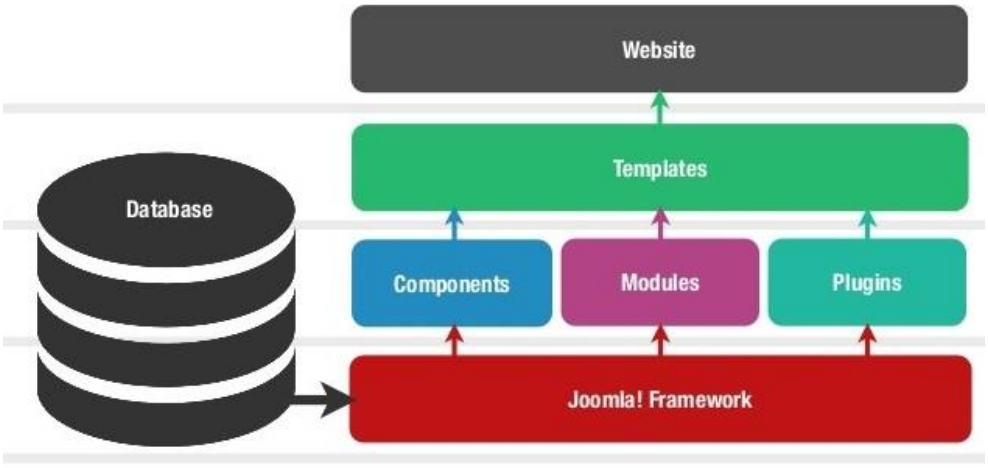
Također novom strategijom određeno je da će sve verzije Joomle biti kompatibilne s prijašnjom verzijom sustava, što znači da će sve ranije značajke sustava raditi i nakon njegove nadogradnje, kao i proširenja razvijena od trećih strana. Ali kako time sustav postaje kompleksniji i time teži za održavanje odlučeno je da se kompatibilnost može narušiti, ali samo izlaskom nove glavne verzije sustava.

3.2 Arhitektura

Joomla sustav za upravljanje web sadržajem izgrađen je na istoimenom PHP programskom okviru koji se može koristi samostalno za izradu web i command line aplikacija u PHP-u. Ovaj programski okvir ne sadrži proširenja i značajke sadržane u Joomla CMS-u već pruža temelj pri izradi aplikacija.

Joomla, kao i proširenja sustava, pisana je u PHP programskom jeziku i koristi načela objektno-orijentiranog programiranja (OOP) i MVC oblikovni obrazac. Podržava MySQL, MS SQL, PostgreSQL baze podataka i Apache, Nginx, Microsoft IIS web servere, a sustav je moguće koristiti na Linux, Microsoft Windows i Apple operacijskim sustavima. Skraćeno, Joomla koristi LAMP, WAMP ili MAMP tehnologije.

U Joomla CMS-u proširenja sustava se dijele na 5 različitih tipova: komponente, moduli, priključci, predlošci i jezici. U dokumentaciji Joomle ovi su tipovi opisani narednim tekstom.



Slika 2 Arhitektura Joomla CMS-a (Czysz, 2014)

Komponente su najveća i najkompleksnija proširenja u Joomli te su većinom vezana uz stavke iz glavnog izbornika. One su glavne funkcijske jedinice sustava i mogu se promatrati kao male aplikacije unutar sustav. Pri razvoju komponenti u Joomla sustavu nalazimo dvije varijante, prva u kojoj komponente prate MVC uzorak i druga gdje komponenta pri pozivu vraća HTML kod.

Većina komponenti se dijeli na administratorski i front-end dio. Prvi dio korisniku sustava pruža sučelje za konfiguraciju i upravljanje komponentom te se do njega dolazi u administracijskoj aplikaciji Joomla sustava, dok je zadaća drugog dijela prikazati komponentu na stranici nakon što je posjetitelj zatraži.

Moduli su jednostavna proširenja koja se koriste u prikazu stranice, te su pozicionirani oko komponente. Mogu biti vezani uz stavku iz glavnog izbornika, kao i uz komponentu iz kojih može dohvaćati određene podatke. Ova povezanost nije uvjet za postojanje modula već on može biti i statički HTML ili text koji korisniku pruža informacije ili služi kao navigacija. Modulima je moguće upravljati pomoću Module Manager opcije koja se nalazi unutar administracijske aplikacije sustava, a kako moduli mogu biti samostalne jedinice unutar stranice, jedan modul je moguće koristiti na svim ili samo nekim stranicama.

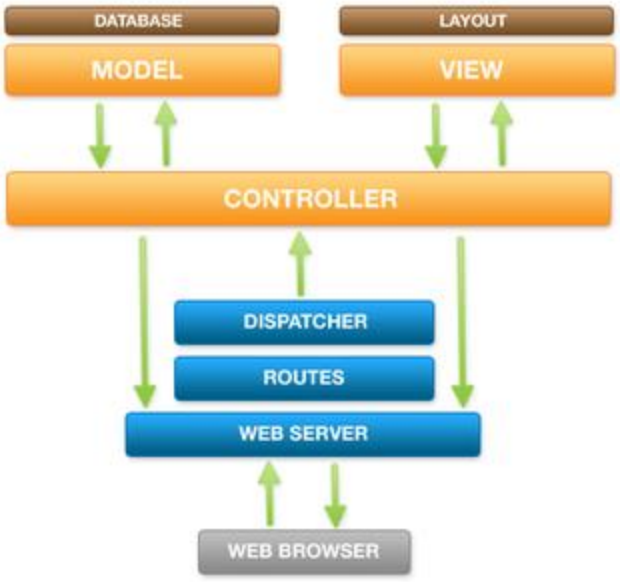
Priključci (plugins) su proširenja sustava koji su u svojoj suštini event handleri te se mogu aktivirati izvođenjem bilo kojeg dijela sustava. Sustav priključaka razvijen je na principu uzorka promatrača (engl. *observer pattern*) u kojem promatrani objekt, JDispatcher, sadrži listu promatrača koje obavještava o promjenama pozivom jedne od promatračevih metoda,

kod Joomla CMS-a objekt promatrač proširuje JPlugin klasu. Jedan od primjera priključka je presretanje komentara korisnika i filtriranje zabranjenih riječi prije objave komentara.

Predlošci (templates) definiraju dizajn web stranice. Sadrže pozicije na kojim se prikazuju jedna komponenta i proizvoljan broj modula. Osim front end predložaka koji se sadržaj prezentira krajnjem korisniku u Joomla CMS-u se koriste i back end predlošci koji se mijenja dizajn administratorske aplikacije sustava.

Jezici su najjednostavnija proširenja sustava. Bez obzira da li je jezik uključen u jezgru[7] sustava ili je dodan kao proširenje nalazi se u datoteci koja sadrži ključ/vrijednost parove koji omogućavaju prevođenje statičkog teksta unutar sustava. Paketi u kojima su sadržani jezici utječu na front end i administratorski dio sustav, odnosno moguće je prevesti tekst na cijelom sustavu.

Kao što je i ranije rečeno Joomla CMS (kao i programski okvir) se zasniva na MVC-u koji je u sustavu implementiran korištenjem JModel, JView i JController klasa. Tijek obrade zahtjeva dobivenog od strane korisnika opisan je u nastavku.



Slika 3 Joomla model-view-controller (MVC) (Sharma)

Nakon zaprimanja zahtjeva od strane korisnika, bez obzira radilo se o dohvaćanju ili slanju podataka, pokreće se njegova obrade. Prvi korak obrade je analiza URL-a da bi se utvrdilo koja je komponenta zadužena za njegovu obradu, a zatim i sama predaja kontrole komponenti.

[7] Jezgrom (engl. *core*) Joomla CMS-a naziva se sustav koji ne sadrži proširenja i značajke koje nisu uključene u instalacijski paket.

Kako je upravljač (engl. *controller*) zadužen za analizu zahtjeva i odabir modela, kako bi se vratio odgovarajući rezultat na korisnikov zahtjev, komponenta prosljeđuje kontrolu upravljaču. U aplikaciji kod predaje kontrole komponenta dohvaća instancu upravljača koji joj je pridružen te pokreće izvođenje koda upravljača.

Također, upravljač je zadužen za odabir odgovarajućeg pogleda (engl. *view*) i prosljeđivanje podataka iz modela u pogled kako bi se rezultat prikazao korisniku.

Model sadrži podatke koji se koriste u komponentama, te se u većini slučajeva ti podatci dohvaćaju iz baze podataka ali mogu biti smješteni i na drugim izvorima kao što je API neke druge web aplikacije. Osim dohvaćanja podataka model je zadužen za spremanje i ažuriranje podataka u bazi.

Pogled je zadužen za kreiranje izlaznog rezultata koji se korisniku prikazuje preko komponente u pregledniku. Kod kreiranja pogleda šalje se zahtjev za dohvaćanje podatka iz modela. Nakon što pogled dobije tražene podatke prosljeđuje ih u predložak koji ih formatira na odgovarajući način.

Kada je izlazni rezultat kreiran u pogledu kontrola izvođenja se prosljeđuje u programski okvir koji učitava i izvršava predložak u kojem se kombiniraju izlazni rezultat komponente i aktivnih modula na toj stranici. Ovo spajanje svih elemenata stranice prije prosljeđivanja na klijentsku stranu napravljeno je iz razloga da bi se u pregledniku korisniku sadržaj dostavio kao jedna stranica.

Predlošci su odvojeni od pregleda zbog toga što se time omogućava dizajniranje izgleda stranice bez potrebe da se mijenja kod unutar pregleda. Pregled ustvari prosljeđuje podatke, dobivene iz modela, u predložak koji je zadužen za formatiranje i prezentaciju sadržaja korisniku.

4. Usporedba verzija

Prije opisa procedure instalacije, detaljne analize strukture i tijeka izvođenja pojedine verzije Joomla CMS-a dana je kratka usporedba tehničkih značajki i standarda koje koriste ove verzije kako bi dobili opću sliku razlika između njih.

Pregled minimalnih tehničkih uvjeta za pokretanje Joomla web sjedišta prikazani su u sljedećoj tablici:

Tablica 2 Minimalni tehnički uvjeti

Software	Joomla! 2.5	Joomla! 3.x
PHP	v. 5.2.4	v. 5.3.10
Baze podataka:		
MySQL SQL server PostgreSQL	v. 5.0.4 v. 10.50.1600.1 nije podržano	v. 5.1 v. 10.50.1600.1 v. 8.3.18
Web serveri:		
Apache Nginx Microsoft IIS	v. 2.0 v. 1.0 v. 7	v. 2.0 v. 1.0 v. 7

Responzivnost klijentske web stranice i administratorskog dijela predstavljena je u Joomla! 3.0 sustavu te je uključena u jezgru sustava koristeći Bootstrap[8] biblioteku. Ova verziji sustava dolazi s responzivnim predlošcima za klijentsku stranu (Beez3 i Protostar) i administratorsko sučelje (Isis).

Iako jezgra sustava Joomla! 2.5 ne sadrži elemente responzivnost moguće je putem proširenja i konfiguracije sustava dodati podršku za Bootstrap.

Kao format za definiranje i primjenu stilova na predlošcima Joomla 2.5 koriste se datoteke u CSS formatu, dok je u Joomla 3.0 sustavu integriran LESS[9] (engl. *Leaner Style Sheets*) čijim se kompajliranjem generiraju CSS datoteke.

Joomla platforma 12.x predstavljena u Joomla 3.0 je u odnosu na prijašnju verziju 11.x koja se koristila u Joomla 2.5 verziji pročišćena od klasa i metoda koje su stavljene van upotrebe.

[8] Besplatna i open-source front-end web biblioteka za izradu responzivnih web stranica baziran na HTML, CSS i JavaScript-u
[9] Jezično proširenje CSS stilskog jezika

Platforma se koristila do izlaska Joomla 3.1 kada je projekt razvoja platforme obustavljen. Platforma je sjedinjena s Joomla CMS-om te je zamijenjena programskim okvirom Joomla koji se koristi za razvoj Joomla aplikacija.

Do promjene je došlo i kod JavaScript biblioteke koja se koristi u sustavu pa je MooTools zamijenjen s jQuery kao primarnom JS bibliotekom sustava.

U Joomla 3.0 predstavljen je i JUI (engl. *Joomla User Interface*) biblioteka koja olakšava i ubrzava razvoj ekstenzija tako što pruža set alata potreban za razvoj komponenti, a u isto vrijeme ubrzava učitavanje Joomla stranica kada korisnik prelazi s jednog Joomla sjedišta na drugo zbog toga što su JS i CSS datoteke već učitane u korisnikovom pregledniku.

Iako ova stavka ne spada pod tehničke specifikacije ali razlika između sustava je vidljiva već pri samoj instalaciji. Kod Joomla 3.0 sustava proces instalacije je sačinjen od 3 koraka, dok je u prethodnoj verziji ovaj proces bio podijeljen u 7 koraka. Proces instalacije za svaku pojedinu verziju detaljnije je opisan u sljedećim poglavljima.

Obje verzije sustava koje se analiziraju u nastavku ovog rada instalirane su lokalno na računalu. Da bi sustavi ispravno radili na računalu instaliran je softverski paket za Microsoft Windows operacijski sustav pod nazivom WampServer. Korištena je inačica softvera za 64 bitne operacijske sustave, verzija 3.1.7.

Ovaj paket sadrži svu potrebnu softversku podršku za rad s Joomla CMS-om, a to su:

- Apache 2.4.37
- PHP 5.6.40 i 7.2.14[10]
- MySQL 5.7.24

Prije same instalacije oba sustava kreirana je prazna MySQL baza podataka, pomoću phpMyAdmin, koja je potrebna pri instalaciji.

4.1 Joomla 2.5 serija

Kao verzija za analizu sustava odabrana je Joomla! 2.5.28 kao posljednja verzija iz serije 2.5.

4.1.1 Instalacija

Instalacijski paket za sustav preuzet je sa službenih stranica Joomla CMS koja se nalazi u vlasništvu neprofitne organizacije Open Source Matters.

[10] Podrška za PHP 7 uvedena je izlaskom Joomla 3.5 sustava

Preuzeta je datoteka u zip formatu veličine 7,73MB te je zatim premještena i raspakirana na primarnom disku računala unutar WampServer direktorija. Točnije, u www direktoriju iz kojeg će se pokretati Joomla stranice. Nakon raspakiravanja veličina direktorija sustava iznosi 18,1MB.

Da bi započeli instalaciju, u pregledniku je potrebno upisati putanju do direktorija u kojem se nalazi Joomla, odnosno localhost/naziv direktorija, nakon čega se otvara sučelje za instalaciju.

Kao što je prethodno navedeno proces instalacije Joomla 2.5 sustava sastoji se od sedam koraka.

U prvom koraku korisnik odabire jezik koji će se koristiti u nastavku instalacije, a odabirom gumba dalje prelazi na sljedeći korak.

Drugi korak služi za provjeru softvera i postavki prije instalacije sustava. Ovaj korak je podijeljen u dva dijela, obvezne i preporučene postavke. Kod prvih, u slučaju da neki od uvjeta nije zadovoljen postoji mogućnost da se proces instalacije ne izvrši na ispravan način, te je preporučeno prekinuti instalaciju dok sve stavke nisu zadovoljene. Dok, u dijelu preporučenih postavki, u slučaju da nisu zadovoljeni svi uvjeti, Joomla sustav će i dalje ispravno raditi.

U trećem koraku korisniku je prikazan tekst s uvjetima i odredbama GNU opće javne licence.

Kako Joomla stranice sve podatke spremaju u bazu podataka u četvrtkom koraku potrebno je napraviti konfiguraciju baze, odnosno u ovom koraku Joomla CMS se spaja s prethodno kreiranom MySQL bazom, a osim ove baze moguće je koristiti i MS SQL.

U četvrtom koraku odabiremo tip baze podataka (MySQL ili MySQLi API), unosi se ime servera koje je najčešće localhost, korisničko ime i lozinku koji se koriste za pristup bazi, te ime baze podataka i prefiks tablice koji se sastoji od alfanumeričkih znakova i završava donjom crtom. Također je moguće napraviti sigurnosnu kopiju ili izbrisati bazu podataka prethodno instaliranog Joomla sustava.

U slučaju da baza nije prethodno kreirana, sustav prilikom instalacije sam kreira bazu i pri tome koristi podatke unesene u formu.

Peti korak instalacije je opcionalan i može se preskočiti, a uključuje konfiguraciju FTP[11]-a (engl. *File Transfer Protocol*) gdje korisnik unosi osnovne podatke: FTP korisničko ime,

[11] Standardni mrežni protokol koji se koristi za prijenos datoteka između dva računala putem TCP/IP mreže

lozinku, ishodišni direktorij, a u padajućem izborniku je ponuđen izbor da li želi omogućiti FTP sloj, te gumb za automatsko traženje putanje do direktorija i provjeru postavki. Osim osnovnih, korisniku je dana mogućnost unosa naprednih postavki gdje unosi IP adresu i port FTP-a.

Kao što piše i u napomeni prilikom instalacije, kod Windows operacijskog sustava FTP sloj nije potreban. Također se preporučuje da se ne unose podaci o korisničkom imenu i lozinki, već da se oni unose prije svakog prijenosa podataka.

U šestom koraku unosi se osnovni podaci o web sjedištu, računu administratora te je ponuđena mogućnost instalacije uzorka podataka na stranici.

Kod unosa podataka za web sjedište potrebno je navesti ime sjedišta, a opcionalno se može unijeti metapodatke koje će koristiti tražilice i sjedište staviti u status neaktivno. U slučaju da je sjedište neaktivno pri pokušaju pristupa sjedištu, u pregledniku se prikazuje obavijest da je stranica u izradi i forma za prijavu.

Kod unosa podataka za račun administratora od korisnika se traži unos valjane e-mail adrese, korisničkog imena i lozinke koji će se koristiti za pristup administratorskom kontrolnom panelu. Ove podatke je moguće promijeniti nakon instalacije kroz administratorsko sučelje Joomla CMS-a.

Instalacijom uzorka podataka korisnik dobiva sadržaj, module i stavke izbornika. Ovom instalacijom struktura Joomla direktorija se ne mijenja već se ovi podaci spremaju u bazu, a ako korisnik prvi puta instalira Joomla sustav, preporučuje se korištenje ovog uzorka kako bi korisnik dobio pregled svih aspekata sustava.

U zadnjem koraku potrebno je izbrisati Joomla instalacijski direktorij iz sigurnosnih razloga, te kako bi mogli dovršiti instalaciju sustava. Direktorij je moguće obrisati pritiskom na gumb u sučelju za instalaciju ili ručno tako da otvorimo Joomla direktorij, koji smo smjestili unutar www direktorija u WampServeru, i obrišemo direktorij pod nazivom installation. Nakon brisanja direktorija moguće je pristupiti sjedištu i administratorskom dijelu sustava putem sučelja za instalaciju ili unosom adrese sjedišta u preglednik. Adresa sjedišta je localhost/naziv_sjedišta dok za administratorsko sučelje unosimo localhost/naziv_sjedišta/administrator.

4.1.2 Struktura direktorija sustava

Direktorij Joomla sustava sastoji se od:

- 4930 datoteka i 1141 direktorija prije instalacije
- 4614 datoteka i 1040 direktorija nakon instalacije

4.1.2.1 Cache direktorij

Kao i većina drugih aplikacija i Joomla koristi privremenu memoriju. Kako kod svakog zahtjeva za dohvaćanje stranice sustav ne bi morao kreirati stranicu i raditi upit u bazu podataka radi vraćanja podataka potrebnih za prikaz, može se omogućiti korištenje privremene memorije. Jednom kada se stranica generira, sprema se unutar direktorija cache te se tamo nalazi određeni vremenski period koji je unaprijed definiran od strane administratora sustava. Kod svakog zahtjeva za dohvaćanje stranice sustav prvo provjerava da li se njezina kopija već nalazi unutar ovog direktorija. Ako je sustav pronašao stranicu čiji rok trajanja nije istekao, korisniku prikazuje kopiju stranice i ne izvodi proces generiranja nove. Korištenje privremene memorije korisno je iz više razloga, kao prvo štede se računalni resursi servera, a kao drugo poboljšano je iskustvo krajnjeg korisnika smanjenjem vremena odaziva stranice.

4.1.2.2 Components direktorij

Ovaj direktorij sadrži komponente koje su uključene u sustav, a kao što je prije rečeno, komponente su male aplikacije unutar sustava koje mogu imati administratorski i klijentski dio, točnije rečeno ovaj direktorij sadrži front-end ili klijentski dio komponente.

Svi direktoriji komponenti koriste isto pravilo pri imenovanju, odnosno ime direktorija započinju sa com_ prefiksom nakon čega dolazi naziv komponente.

Sve komponente koje se nalaze u jezgri Joomla CMS-a koriste MVC uzorak, što znači da sadrže upravljač, pregled i model skripte unutar direktorija u kojem se nalaze. Na primjeru com_contents komponente opisana je namjena PHP datoteka i direktorija koje sadrži:

- Controllers direktorij sadrži article.php datoteku koja je zadužena za izvođenje CRUD operacija (proširuje JControllerForm)
- Helpers direktorij sadrži skripte s metodama koje se koriste u komponenti a ne pripadaju ni jednom dijelu MVC-a ili se koriste na više različitih mjesta
- Models direktorij sadrži PHP datoteke koje predstavljaju modele i direktorij forms u kojem se nalazi article.xml koji definira polja forme za unos ili izmjenu članka
- Views direktorij sadrži poglede koji su potrebni komponenti, a svaki poddirektorij ima view.html.php skriptu s klasom koja proširuje JViewLegacy i direktorij tmpl u kojem se nalaze skripte za generiranje HTML koda komponente

- content.php je ulazna točka komponente, odnosno ovo je prva skripta koja se poziva i zatim kreira i pokreće upravljač komponente
- controller.php je glavni upravljač zadužen za prikaz pregleda korisniku (proširuje JControllerLegacy)
- metadata.xml sadrži metapodatke komponente
- router.php skripta zadužena za kreiranje i parsiranje URL-a

Većina komponenti u ovom direktoriju povezana je sa stavkama iz izbornika, osim komponenti koje se nalaze u com_banners, com_mailto i com_media direktorijima.

Banner komponenta se smješta unutar modula na stranici a zadaća joj je dohvatiti sliku bannera, preusmjeriti korisnika na prethodno definirani URL i povećati brojač klikova na određeni banner.

Mailto komponenta prikazuje formu s poljima za unos podataka za slanje ako korisnik odluči poslati članak putem elektroničke pošte.

Media komponenta učitava Media Manager kada korisnik u front-endu želi umetnuti sliku unutar članka.

4.1.2.3 Includes direktorij

U ovom direktoriju se nalaze skripte za učitavanje konstanti i klasa potrebnih za pokretanje sustava, a u nastavku su opisane datoteke koje se nalaze unutar njega.

Application datoteka sadrži JSite klasu koja je potklasa JApplication klase. Ova klasa je ujedno i front-end aplikacija koju korisnik koristi kada dođe na web sjedište, a zadužena je za inicijalizaciju korisnika i jezičnih postavki, prijavu i autorizaciju korisnika, dohvaćanje i prikaz elemenata stranice, te preusmjeravanje korisnika na određeni URL.

Defines datoteka sadrži definiciju globalnih varijabli koje sadrže putanje do direktorija koji se koriste unutar sustava.

Framework datoteka sadrži kod koji provjerava da li se u glavnom direktoriju sustava ne nalazi configuration.php datoteka i da li se unutar instalacijskog direktorija nalazi index.php. U slučaju da je ova provjera zadovoljena korisnika se preusmjerava na instalacijski proces te se prekida izvođenje ove skripte. Ako provjera nije zadovoljena, učitavaju se konfiguracija i dijelovi programskog okvira potrebni za daljnje izvođenje aplikacije.

Menu datoteka služi za dohvaćanje stavki glavnog izbornika sjedišta iz baze podataka.

Pathway datoteka sadrži JPathwaySite koja je potklasa JPathway klase. Ova klasa sadrži konstruktor koji kreira putanju do mjesta na kojem se korisnik nalazi. Jedna od primjena ove klase je unutar modula mod_breadcrumbs.

Router datoteka je zadužena za kreiranje i parsiranje URI-a, također sadrži i metodu za kreiranje SEF (engl. *Search engine friendly*) putanja radi lakšeg indeksiranja stranica.

4.1.2.4 Language direktorij

Svi jezici instalirani unutar sustava nalaze se na ovom mjestu. Iako nakon same instalacije ovaj direktorij sadrži samo datoteke s konstantama na engleskom jeziku, moguće je instalirati jedan od 62 podržana jezika ili učitati jezični paket. Obje metode postavljanja novog jezika nalaze se u administracijskom dijelu aplikacije unutar language managera u extensions meniju.

4.1.2.5 Modules direktorij

Ovaj direktorij sadrži sve front-end module koji su instalirani na sustavu, a svaki se modul nalazi u svom poddirektoriju. Nazivi ovih direktorija prate isto pravilo imenovanja kao i komponente uz razliku da je prefiks u nazivu mod_ nakon kojeg dolazi ime modula. Struktura većine direktorija modula je ista, a za primjer je uzet modul articles_popular koji sadrži sljedeće datoteke i poddirektorij:

- Tmpl direktorij u kojem se nalaze default.php skripta za generiranje HTML koda komponente
- helper.php skripta dohvaća podatke koji se koriste u modulu
- mod_articles_popular.php je ulazna točka modula
- mod_articles_popular.xml sadrži opis modula, popis datoteka i direktorija, jezika i dodatne konfiguraciju parametara modula

4.1.2.6 Templates direktorij

Ovaj direktorij sadrži poddirektorije predložaka koji su instalirani unutar sustava, te su namijenjeni korištenju na klijentskoj aplikaciji. Svaki od ovih direktorija može sadržavati sljedeće poddirektorije:

- css sadrži datoteke istoimenog formata koje se koriste unutar predloška
- html sadrži php datoteke kojima je moguće pregaziti postavke sustava za prikaza elemenata na predlošku

- javascript sadrži datoteke istoimenog formata kojima se proširuju funkcionalnosti na predlošku
- font sadrži datoteke u kojima se nalazi definicija fonta
- images sadrži slike koje se koriste u predlošku
- languages sadrži datoteke u kojima se nalazi prijevod konstanti koje se prikazuju na predlošku

Ovaj direktorij također sadrži i datoteke:

- component.php sadrži predložak koji se koristi za prikaz članka ili forme za slanje e-mail poruke kada korisnik odluči ispisati članak putem printera ili poslati članak putem e-mail poruke
- error.php sadrži predložak koji se koristi kod ispisa greške korisniku
- favicon je ikona koja se prikazuje kao logo kartice stranice u web pretraživaču
- index.php sadrži predložak stranice koji određuje poziciju prikaza grešaka, modula i komponenti
- template_preview je slika prikaza stranice
- template_thumbnail je ista kao i prethodna slika samo manjih dimenzija
- templateDetails.xml je datoteka koja sadrži popis direktorija, datoteka, jezika, pozicija na predlošku, te konfiguraciju parametara predloška

Osim instaliranih predložaka, templates sadrži system direktorij koji služi kao rezerva u slučaju kada neki od dijelova predloška ne funkcionira pravilno.

4.1.2.7 Administrator direktorij

Kao što je i ranije navedeno Joomla se sastoji od dvije zasebne aplikacije, front-end koji se prikazuje korisniku u obliku web stranice i back-end kojeg čini administratorska aplikacija pohranjena unutar ovog direktorija.

Kao i u početnom direktoriju sustava, unutar administrator direktorija se također nalaze cache, components, includes, language, modules i templates poddirektoriji koji su prethodno opisani, a osim navedenih poddirektorija, administracija sadrži help i manifest direktorije.

Unutar help direktorij nalaze se helpsite.xml datoteka s poveznicom na Joomla dokumentaciju i poddirektorij HTML datoteka s opisom i uputama za korištenje komponenti i značajki sustava. Datoteke unutar poddirektorija s uputama korisnika preusmjeravaju na Joomla! Help web sjedište, ali ih je moguće instalirati i lokalno na serveru.

Manifest direktorij sadrži tri poddirektorija pod nazivima files, libraries i packages.

Prvi, files direktorij, sadrži joomla.xml datoteku u kojoj se nalazi popis svih direktorija i datoteka u početnom Joomla direktoriju, poveznice na servere za ažuriranje i putanje potrebne za ažuriranje sheme baze podataka.

Drugi, libraries direktorij, sadrži XML datoteku za svaku biblioteku koja se koristi u sustavu. Unutar nje je naveden popis svih datoteka i direktorija koji se nalaze unutar libraries direktorija u početnom Joomla direktoriju, kao i poveznica na stranicu za njihovo ažuriranje.

Treći, packages direktorij, se koristi kako bi se korisniku olakšala instalacija i brisanje više povezanih proširenja sustava. Na primjer, kada korisnik želi instalirati više povezanih proširenja, to radi na način da instalira cijeli paket proširenja a ne svako proširenje zasebno, a isto vrijedi i za postupak brisanja.

4.1.2.8 Libraries

Ovaj direktorij sadrži biblioteke koje sustav koristi, a te biblioteke možemo podijeliti na vanjske koje su razvijene od treće strane i Joomla biblioteke.

Unutar libraries direktorija nalaze se dvije vanjske biblioteke i dva programski okvir. PHPMailer je biblioteka koja se koristi za slanje elektroničke pošte, phputf8 je biblioteka koja se koristi za obradu i manipulaciju UTF-8 teksta. Programski okvir SimplePie se koristi kod funkcionalnosti RSS-a i Atomic, dok se PHPass upotrebljava za hashiranje i provjeru lozinke.

Od Joomla biblioteka i programskog okvira unutar libraries se nalaze Joomla platforma koja je smještena u Joomla direktoriju, dok se klase koje nisu sadržane u platformi a važne su za funkcioniranje sustava nalaze unutar direktorija cms.

Osim poddirektorija libraries sadrži i četiri PHP datoteke:

- cms – skripta kreira konstantu putanje do platforme, kreira konstantu verzije Joomla sustava i učitava JLoader klasu
- import – skripta kreira konstante i učitava osnovne klase platforme
- loader – sadrži metode za učitavanje klase
- platform – sadrži informacija o platformi i metodu za provjeru verzije PHP-a

4.1.2.9 Installation direktorij

Razlika u broju direktorija navedena na početku odlomka Struktura direktorija sustava javlja se zato što prije instalacije Joomla direktorij sadrži instalacijski (engl. *installation*) direktorij

koji, kao što i ime kaže, služi za instalaciju sustava. Ovaj direktorij, kao i administration, možemo promatrati kao zasebnu aplikaciju, iako administracijska aplikacija i instalacijski proces koriste određeni programski kod koji se nalazi u početnom Joomla direktoriju.

Unutar ovog direktoriju se nalaze includes, language i template direktorij koji su sadržani i u početnom Joomla direktoriju, a svrha i sadržaja ovih direktorija već je prethodno opisan.

Također, osim gore navedenih poddirektorija, unutar instalacijskog direktorija se nalaze i sljedeći poddirektoriji:

- controllers sadrži upravljača koji se koristi tijekom instalacije
- helpers sadrži database.php unutar koje se nalazi metoda za dohvaćanje JDatabase objekt i html poddirektorij sa installation.php datotekom čija je zadaća u bočnoj navigaciji označiti trenutačan korak u procesu instalacije
- models sadrži modele koji se koriste prilikom instalacije
- sql sadrži SQL skripte za kreiranje i ažuriranje osnovnih tablica koje sustav koristi, kao i tri skripte za punjenje baze s oglednim sadržajem
- views sadrži poglede koji su potrebni pri instalaciji, a svaki korak instalacijskog procesa ima svoj poddirektorij unutar kojeg se nalazi view.html.php datoteka s metodom za prikaz pogleda i tmpl direktorij s HTML kodom stranice

Osim direktorija unutar instalacijske aplikacije nalaze se i datoteke:

- index.php datoteka čijim se izvođenjem pokreće instalacijski proces sustava.
- controller.php odnosno primarni upravljač instalacije
- configuration.php-dist koja se koristi pri ručnoj instalacije sustava kada se ne koristi web preglednik
- CHANGELOG je dnevnik promjena na sustavu
- COPYRIGHT sadrži tekst autorskih prava (GNU General Public License)
- INSTALL sadrži upute za instalaciju
- LICENSES sadrži popis licenci koje se koriste unutar sustava i njihov puni tekst
- localise.xml datoteka se koristi kako bi se postavio zadani jezik i ogledni podaci u procesu instalacije, odnosno unutar forceLang oznake postavlja se jezik koji će biti odabran kao zadani jezik instalacije dok se za ogledne podatke koristi sampledata oznaka

4.1.2.10 Cli direktorij

Unutar cli (engl. *command-line-interface*) direktorija nalaze se aplikacije koje se izvršavaju pokretanjem pomoću komandne linije. Ovaj direktorij sadrži tri PHP skripte od kojih dvije u nazivu sadrže oznaku CRON poslova dok zadnja, finder_indexer.php datoteka, služi za pokretanje Smart Search indeksacije. Update_cron skripta je namijenjena za dohvaćanje informacija o ažuriranju za sva proširenja i pohranjuje ih u bazu podataka, dok garbagecron iz privremene memorije briše one podatke čije je vrijeme trajanja isteklo.

4.1.2.11 Images direktorij

Sve slike prikazane na web sjedištu nalaze se unutar ovog direktorija, te mogu biti smještene i u poddirektorije kao što su banners u kojem se nalaze slike koje se prikazuju unutar modula banner. Ako se korisnik pri instalaciji odluči instalirati ogledne primjere stranica, kreira se i direktorij sampledata koja sadrži slike koje se prikazuju na tim stranicama.

4.1.2.12 Logs direktorij

U ovom direktoriju se spremaju informacije o aktivnosti sustava i njegovih proširenja, a primjer jedne od njih je zapisivanje grešaka koje su se dogodile tijekom izvođenja, te se one upisuju u error.php datoteku.

4.1.2.13 Media direktorij

Unutar media direktorija nalaze se CSS i JavaScript datoteke koje se koriste unutar sustava. Osim njih ovdje su spremljene i slike, točnije ikone koje se prikazuju u sustavu. Također, ranije spomenuta MooTools biblioteka smještena je unutar system poddirektorija.

4.1.2.14 Plugins direktorij

Svi priključci Joomla CMS-a proširuju klasu JPlugin, te se prilikom instalacije smještaju unutar ovog direktorija. Sve klase priključaka imaju isti način imenovanja, prvi dio imena sadrži plg što označava da se radi o klasi priključka, zatim ime direktorija u kojem se nalaze i na kraju ime PHP datoteke unutar koje se priključak nalazi. Uz PHP datoteku u svakom direktoriju priključka nalazi se i XML datoteka koja sadrži općenite informacije o priključku i njegovom autoru, popis datoteka koje priključak koristi, popis jezika za lokalizaciju i dodatnih parametrara priključka koje je moguće promijeniti putem Plugin managera u administracijskoj aplikaciji. Ovisno o svojoj zadaći i događajima koji ih aktiviraju priključci su svrstani u jedan od sljedećih direktorija.

Authentication direktorij sadrži priključke koji se aktiviraju prilikom prijave pozivom onUserAuthenticate metode koja provjerava korisničko ime i lozinku korisnika. Jezgra Joomla sustava sadrži tri različite varijante autentifikacijskih priključaka, a to su plgAuthenticationGMail, plgAuthenticationJoomla i plgAuthenticationLdap.

Captcha direktorij sadrži besplatan servis razvijen od strane Google-a, odnosno reCAPTCHA priključak čija je zadaća zaštiti sjedište od botova, odnosno utvrditi da je korisnik čovjek tako što od njega traži unos teksta sa slike koja prikazuje skup iskrivljenih slova ili brojeva. Ovaj priključak implementira tri događaja: onInit, onDisplay i onCheckAnswer.

Content direktorij sadrži priključke i događaje koji se aktiviraju kod kreiranja i uređivanja svih vrsta sadržaja.

Editors direktorij sadrži priključke i događaje vezane uz uređivače teksta, te ovi priključci sadrže sljedeće događaje:

- onInit – inicijalizira uređivač
- onGetContent – dohvaća sadržaj
- onSetContent – postavlja sadržaj
- onSave – kopira sadržaj
- onGetInsertMethod – dohvaća metodu za postavljanje sadržaja u urednik
- onDisplay – prikazuje uređivač
- _displayButtons – prikazuje gumbe uređivača

Editors-xtd priključci dodaju gumbe koji se nalaze ispod uređivača teksta, a jezgra Joomla sustava dolazi s gumbima za dodavanje članka, slike, prijeloma stranice i opcije pročitaj više unutar sadržaja koji se nalazi u uređivaču.

Extensions direktorij sadrži priključak koji se aktivira na događaje vezane uz proširenja sustava, točnije ovaj priključak se aktivira nakon instalacije, uklanjanja i ažuriranja proširenja. Također sadrži i implementaciju događaja koji dodaje poveznicu na stranicu s ažuriranjima za proširenje u bazu podataka.

Finder direktorij sadrži priključke koji služe za indeksiranje sadržaja za korištenje Smart Search-a. Indeksiranje je moguće provesti nad kategorijama, kontaktima, sadržajem, vijestima i web linkovima.

sadrži priključke koji sadrže metodu za provjeru određenih uvjeta, onGetIcons, te korisnika obavještavaju o dostupnim nadogradnjama sustava ili njegovih proširenja, dok PlgQuickiconEosnotify priključak korisnika upozorava o prestanku potpore instalirane verzije sustava.

Search direktorij sadrži priključke koji implementiraju dvije metode, onContentSearchAreas stvara niz stavaka sadržaja koji će se pretraživati dok onContentSearch metoda izvršava upit nad bazom, odnosno traži komponente. Jezgra Joomla sustava sadrži priključke za pretragu kategorija, kontakata, sadržaja, vijesti i web linkova.

System direktorij sadrži priključke koji se aktiviraju u tijeku izvođenja sustava.

User direktorij sadrži priključke koji se aktiviraju na događaje vezane uz aktivnosti kod kojih se podaci korisnika mijenjaju, odnosno spremanja i brisanja podataka, i prilikom prijave i odjave korisnika sa sustava.

4.1.2.15 Tmp direktorij

Ovaj direktorij koriste sustav i njegova proširenja kada je potrebno privremeno pohraniti neke podatke prilikom instalacije ili tijekom obrade datoteka.

4.1.2.16 Datoteke

Iz sigurnosnih razloga svaki poddirektorij sadrži index.html datoteku zato što kada korisnik upiše putanju do nekog direktorija unutar Joomla sustava web server traži index.php ili index.html datoteku za prikaz. U slučaju da datoteka nije pronađena, ispisuje se lista svih datoteka i direktorija koji se nalaze na unesenoj putanji, a ovakav uvid u sustav smatra se sigurnosnim propustom.

Prethodna tvrdnja se ne odnosi na administrator, installation i početni direktorij sustava je se u njima nalazi index.php datoteka koja služi kao ulazna točka kod pokretanja aplikacija.

Osim poddirektorija početni Joomla direktorij sadrži i određene datoteke:

- configuration.php sadrži osnovne postavke sustava, a kreira se nakon instalacije sustava
- htaccess.txt je moguće preimenovati u .htaccess i koristiti unutar Apache web servera kako bi se uklonilo index.php iz svih Joomla URL-a
- index.php je ulazna točka u front-end aplikaciju sustava

- joomla.xml sadrži osnovne informacije o sustavu, putanju do direktorija s ažuriranjima za podržane baze podataka, popis direktorija i datoteka u glavnom direktoriju sustava, poveznicu na servere za ažuriranje sustava i putanju do script.php datoteke koja sadrži metodu za ažuriranje sustava
- LICENSE.txt sadrži tekst GNU General Public License
- README.txt sadrži općenite informacije o Joomli
- robots.txt sprječava pristup web robota u poddirektorije sustava
- web.config.txt ova datoteka je po namjeni ista kao i .htaccess ali se koristi unutar Microsoft ISS web servera

4.1.3 Baza podataka

Joomla koristi bazu podataka kako bi pohranila sve podatke koji se nalaze na sjedištu, a Joomla 2.5 verzija sustava podržava MySQL, Microsoft SQL server i PostgreSQL baze podataka.

Baza nakon instalacije sadrži 61 tablicu, a naziv svih tablica započinje prefiksom definiranim tijekom instalacije sustava. Unutar koda sve tablice koriste generički prefiks #__, u kojem se pri izvođenju upita u bazu znak # zamijeni prefiksom koji se nalazi u configuration.php, točnije s vrijednošću varijable $dbprefix.

Bazi podataka se može pristupiti preko programa phpMyAdmin koji omogućava pregled baze, kreiranje i brisanje tablica, manipulaciju i izvoz podataka koji se nalaze u bazi.

Tablice koje se koriste u jezgri Joomla sustava su:

- #__assets - sadrži popis svih komponenti, kategorija i članaka, te sadrži stupac u kojem je definirano koja grupa korisnika ima pravo vršiti određene radnje nad stavkom. Tablica se koristi svaki puta kada se provjerava da li je korisnik ovlašten za izvođenje određene radnje
- #__associations – koristi se u sustavima koji koriste više jezika da bi se određenu stavku izbornika spojilo sa stavkom izbornika na drugom jeziku
- #__banners – popis svih bannera koji su kreirani unutar sustava
- #__banner_clients – podaci o klijentima bannera
- #__banner_tracks – podaci o praćenju bannera, odnosno njihovih pojavljivanja i klikova
- #__categories – popis svih kategorija koje su definirane unutar sustava

- #__contant_details – informacije o svim kontaktima definiranim na sustava
- #__content – članci kreirani unutar sustava
- #__content_frontpage – poredak članaka unutar Featured Articles tipa stavke izbornika, aktivira se na način da se u Layout options pod opcijom Article Order odabere Featured Articles Order
- #__content_rating – sadrži prosječnu sumu i broj ocjena članka
- #__core_log_searches – popis pojmova pretraživanja unutar sjedišta od strane korisnika
- #__extensions – popis proširenja instaliranih unutar sustava
- #__finder_filters – filteri pretraživanja definirani u Smart Search-u
- #__finder_links – stavke sadržaja indeksirane pomoću Smart Search-a
- #__finder_links_terms – ovaj naziv koristi 15 tablica, a razlikuju se po tome što su označene različitim sufiksima (znamenkama od 0 do 9 i slovima od a do f), mapiraju pojmove pretraživanja (#__finder_terms) i stavke iz prethodne tablice, te im se dodjeljuje težinu
- #__finder_taxonomy – popis tipova stavki
- #__finder_taxonomy_map – mapira stavke sadržaja (#__finder_links) s tipovima iz prethodne tablice
- #__finder_terms – sadrži popis svih mogućih indeksiranih pojmova pretraživanja
- #__finder_terms_common – sadrži popis svih riječi koje se učestalo pojavljuju u jeziku, te koje se preskače tijekom indeksacije
- #__finder_tokens – privremena tablica koja se koristi tijekom indeksacije
- #__finder_tokens_aggregate – privremena tablica koja se koristi tijekom indeksacije
- #__finder_types – sadrži popis svih tipova komponenti
- #__languages – jezici instalirani unutar sustava
- #__menu – svi izbornici definirani unutar sustavu
- #__menu_types – popis svih izbornika definiranih na front-end stranici
- #__messages – sadrži sve poruke poslane od strane korisnika unutar sustava
- #__messages_cfg – bilježi promjene postavki u Private Messages Manageru back-end korisnika
- #__modules – popis modula unutar sustava
- #__modules_menu – prikazuje povezanost modula i stavki izbornika, a pod menuid stupcem može sadržavati tri vrste vrijednosti: 0 označava da je modul povezan na sve

stavke, pozitivan broj označava stavku na koju se modul veže, a negativan broj označava da se modul veže na sve stavke osim navedene

- #__newsfeeds – popis svih kreiranih stavki u Components -> Newsfeeds
- #__overrider – sadrži jezična nadjačanja koja se koriste na stranici
- #__redirect_links – sadrži kreirana preusmjeravanja sa starog na novi URL
- #__schemas – sadrži zapis za svako proširenje koje je napravilo izmjenu na bazi tijekom instalacije kao i verziju proširenja
- #__sessions – popis aktivnih sesija
- #__template_styles – predlošci instalirani na front-end i back-end strani sustava
- #__updates – dostupni paketi za instalaciju
- #__update_categories – kategorizacija ažuriranja
- #__update_sites – popis poveznice na stranice s ažuriranjima
- #__update_sites_extensions – povezuje stavke iz tablice proširenja s poveznicama za ažuriranje
- #__usergroups – grupe korisnika definirane unutar sustava
- #__users – popis korisnika sustava
- #__user_notes – sadrži bilješke o korisnicima
- #__user_profiles – sadrži uparenje korisnika sa svakim poljem korisničkog profila koja se popunjavaju u User manageru
- #__user_usergroup_map - svaki red predstavlja korisnika i pridruženu mu grupu
- #__viewlevels - razine korisničkog prikaza
- #__weblinks – sve web poveznice definirane na sustavu

4.1.4 Tijek izvođenja

Kao što je ranije kod analize strukture direktorija navedeno, index.php skripta je ulazna točka sustava. To znači da kada unesemo URL adresu klijentske stranice unutar preglednika poziva se index.php skripta u korijenskom direktoriju sustava, isto se odnosi i na pokretanje administratorske aplikacije uz razliku da se tada ova skripta nalazi unutar administrator direktorija.

Većina PHP datoteka koja se nalazi unutar sustava započinje komentarom unutar kojega je naveden paket i potpaket kojemu ona pripada. Također su navedene informacije o autorskim podacima i licencama koje se odnose na tu datoteku.

```
/**
 * @package    Joomla.Administrator
 * @copyright Copyright (C) 2005 - 2014 Open Source Matters, Inc. All
rights reserved.
 * @license    GNU General Public License version 2 or later; see
LICENSE.txt
 */
```

U nastavku je dan pregled tijeka izvođenja a kako je index.php skripta inicijalna datoteka pri pokretanju počet ćemo s njezinom analizom.

Nakon komentara na samom početku datoteke, nalaze se sljedeće linije koda:

```
// Set flag that this is a parent file
define('_JEXEC', 1);
define('DS', DIRECTORY_SEPARATOR);
```

Prva linija koda definira konstantu _JEXEC te ona označava sigurnu ulaznu točku u sustav. Važno je napomenuti da konstante imaju globalni doseg, odnosno vidljive su unutar cijele aplikacije.

Sve ostale PHP datoteke sadrže *defined('_JEXEC') or die* liniju pomoću koje provjeravaju da li je konstanta definirana a time se osigurava da se toj datoteci pristupilo preko ulazne točke. Također ova metoda maskira grešku, odnosno ako izravno izvršimo PHP datoteku, koja nije index.php, a navedena metoda se ne nalazi na početku ove datoteke, u pregledniku se ispisuje greška s punom putanjom do te datoteke.

Druga linija definira DS konstantu, odnosno znak koji se koristi kao separator između direktorija unutar putanje. Ova konstanta se definira zato što PHP na Windows OS-u koristi „\“, dok kod Linux i Mac OS-a koristi „/“.

U sljedećem dijelu datoteke definiraju se konstante koje sadrže putanje do određenih direktorija sustava, kreiranje konstanti se odvija unutar defines.php skripte.

```
if (file_exists(dirname(__FILE__) . '/defines.php')) {
   include_once dirname(__FILE__) . '/defines.php';
}

if (!defined('_JDEFINES')) {
   define('JPATH_BASE', dirname(__FILE__));
   require_once JPATH_BASE.'/includes/defines.php';
}
```

U prvom if bloku provjerava se da li u korijenskom direktoriju postoji defines.php datoteka, te time sustav nudi mogućnost definiranja vlastite datoteke za definiranje putanja, odnosno moguće je pregaziti zadanu defines.php datoteku uz uvjet da se unutar nje definira _JDEFINES konstanta.

Drugi blok provjerava da li je ta konstanta kreirana, ako nije definira se JPATH_BASE konstanta koja sadrži putanju do korijenskog direktorija sustava. Dohvaćanje putanje roditeljskog direktorija trenutne datoteke vrši se pomoću metode dirname(__FILE__), u kojoj proslijeđeni parametar __FILE__ dohvaća putanju i naziv trenutne datoteke. Nakon kreiranja konstante putanje sustav dohvaća defines.php skriptu unutar includes poddirektorija.

Kao što je vidljivo iz koda, sustav koristi dvije različite metode za dohvaćanje datoteka, include_once i require_once. Obje metode će uključiti kod unutar skripte samo ako sadržaj tih datoteka nije uključen već ranije unutar trenutnog tijeka izvođenja. Jedina razlika je što prva metoda neće zaustaviti izvođenje programa kada datoteka ne postoji dok druga hoće, iz razloga što baca grešku.

Sadržaj defines.php datoteka dan je u nastavku:

```
//Global definitions.
//Joomla framework path definitions.
$parts = explode(DIRECTORY_SEPARATOR, JPATH_BASE);
array_pop($parts);

//Defines.
define('JPATH_ROOT',          implode(DIRECTORY_SEPARATOR, $parts));
define('JPATH_SITE',          JPATH_ROOT);
define('JPATH_CONFIGURATION',  JPATH_ROOT);
define('JPATH_ADMINISTRATOR',  JPATH_ROOT . '/administrator');
define('JPATH_LIBRARIES',      JPATH_ROOT . '/libraries');
define('JPATH_PLUGINS',           JPATH_ROOT . '/plugins');
define('JPATH_INSTALLATION',   JPATH_ROOT . '/installation');
define('JPATH_THEMES',        JPATH_BASE . '/templates');
define('JPATH_CACHE',         JPATH_ROOT . '/cache');
define('JPATH_MANIFESTS',      JPATH_ADMINISTRATOR . '/manifests');
```

U sljedećem koraku sustav učitava Joomla programski okvir pomoću datoteke framework.php koja se nalazi unutar includes poddirektorija.

```
require_once JPATH_BASE.'/includes/framework.php';
```

Unutar skripte programskog okvira provjerava se da li se unutar korijenskog direktorija nalazi configuration.php datoteka i da instalacijski direktorij ne postoji. Ako jedan od ova dva uvjeta nije zadovoljen i sustav pronađe index.php datoteku unutar instalacijskog direktorija tijek izvođenja se preusmjerava na instalacijski proces pomoću header metode koja šalje HTTP zaglavlje, odnosno lokaciju instalacijske index.php datoteke.

Nakon ove provjere slijedi učitavanje import.php skripte koja se nalazi unutar libraries direktorija.

```
// System includes.
require_once JPATH_LIBRARIES.'/import.php';
```

Unutar import.php skripte učitavaju se klase potrebne za daljnji rad sustava i provjerava na kojem se OS-u sustav pokreće. Skripta učitava JPlatform i JLoader klasu pomoću require_once metode, a nakon toga postavlja autoloader[12] koji se nalazi unutar JLoader klase. Postavljanje autoloadera izvršava se pomoću setup metode gdje se pomoću PHP metode spl_autoload_register definiraju load i _autoload metode JLoader klase kao one koje će se koristiti kao autoloader funkcije.

Unutar import.php skripte, učitavaju se i JFactory, JException i JObject klase pomoću import metode.

```
// Import the factory library.
JLoader::import('joomla.factory');

// Import the exception and error handling libraries.
JLoader::import('joomla.error.exception');
// Import the base object library.
JLoader::import('joomla.base.object');
```

Import metoda unutar JLoader klase dodaje ključ i vrijednost u polje classes, gdje je ključ naziv klase kojemu se dodaje prefiks, slovo j, a vrijednost putanju do datoteke unutar koje se klasa nalazi. Osim ove metode koristi se i register metoda koja također dodaje klasu unutar classes polja. Razlika između ovih metoda je ta da import ima ugrađenu provjeru da li je klasa koju se učitava helper klasa.

Povratkom u framework.php skriptu slijedi postavljanje načina obrade grešaka, a zatim se učitava cms.php skripte. Unutar ove skripte registrira se prefiks s putanjom do biblioteka unutar cms poddirektorija koji se nalazi u libraries direktoriju.

```
// Register the library base path for CMS libraries.
JLoader::registerPrefix('J', JPATH_PLATFORM . '/cms');
```

Metoda registerPrefix je još jedna od metoda koja se nalazi u sklopu JLoader klase a vezna je uz autoloader, odnosno kada se pozove _autoload metoda ona prolazi kroz sve registrirane prefikse (polje prefixes) i putanje vezane uz taj prefiks, te unutar tih putanji traži klasu koju se učitava.

Unutar framework.php učitava se i configuration.php datoteka, odnosno JConfig klasa i na temelju podataka iz konfiguracije postavlja se koje će greške sustav ispisivati.

[12] Učitava klase platforme i sustava kada su one potrebne bez da se to eksplicitno navede koristeći jednu od metoda unutar JLoader klase. Potrebno je kreirati novi objekt te se tada koristi autoloder da bi se dohvatilo željenu klasu

Također unutar ove skripte učitavaju se još neke klase koristeći jimport metodu. Ova se metoda nalazi unutar loader.php datoteke i poziva JLoader import metodu.

```
jimport('joomla.application.menu');
jimport('joomla.environment.uri');
jimport('joomla.utilities.utility');
jimport('joomla.event.dispatcher');
jimport('joomla.utilities.arrayhelper');
```

Ovim učitavanjem završava se izvođenje framework.php skripte i vraćamo se na index.php skriptu. Povratkom u skriptu dolazimo na sljedeće linije koda:

```
// Mark afterLoad in the profiler.
JDEBUG ? $_PROFILER->mark('afterLoad') : null;

// Instantiate the application.
$app = JFactory::getApplication('site');

// Initialise the application.
$app->initialise();

// Mark afterIntialise in the profiler.
JDEBUG ? $_PROFILER->mark('afterInitialise') : null;
```

Prva i zadnja linija koda služe za mjerenje performansi sustava i one se kod normalnog pokretanja sustava ne izvršavaju.

Pomoću metode getApplication iz JFactory klase poziva se getInstance metoda unutar JApplication klase koja dohvaća i vraća JSite, JAdministrator ili JInstallation objekt, ovisno o tome kojoj aplikaciji pokušavamo pristupiti. Također u ovom koraku se dohvaća postojeća ili kreira nova sesija, te se učitavaju podaci konfiguracije.

Nakon što je objekt app kreiran poziva se metoda initialise. Unutar ove metode postavlja se jezik koji će se koristiti unutar sustava, a zatim se poziva initialise metoda roditelja, odnosno JApplication klase. Unutar roditeljske metode postavlja se uređivač koji će korisnik koristiti unutar sustava, a nakon toga slijedi učitavanje priključaka koji se nalaze unutar system poddirektorija i pokretanje događaja onAfterInitialise.

```
// Trigger the onAfterInitialise event.
JPluginHelper::importPlugin('system');
$this->triggerEvent('onAfterInitialise');
```

Metoda importPlugin dohvaća system priključke i kreira JDispatcher objekt unutar kojega se spremaju objekti promatrača, odnosno navedeni priključci i njihove metode koje su ujedno i događaji unutar sustava.

Sustav pokreće događaj onAfterInitialise i aktivira plgSystemP3p i plgSystemRemember priključke, odnosno izvršava metodu unutar navedenih klasa koja je istoimena događaju.

Po povratku u index.php skriptu izvršavaju se sljedeće linije koda:

```
// Route the application.
$app->route();

// Mark afterRoute in the profiler.
JDEBUG ? $_PROFILER->mark('afterRoute') : null;
```

Unutar route metode, u prvom koraku poziva se istoimena metoda roditeljske klase, odnosno JApplication klase koja proširuje JSite klasu, te se klonira dohvaćena instanca JUri objekta kako se u nastavku tijekom obrade njegovih podatak ne bi izmijenio postojeći objekt. Unutar ovog objekta, u ovom slučaju, nalazi se URL s putanjom do početne front-end stranice sustava (http://localhost/naziv_stranice/).

```
// Get the full request URI.
$uri = clone JURI::getInstance();

$router = $this->getRouter();
$result = $router->parse($uri);
```

Koristeći metodu parse koja se nalazi u sklopu JRouteSite klase parsira se dobiveni URL, a kao rezultat ova metoda vraća polje s informacijama o početnoj stranici:

- option = "com_content" – ime komponenta koja se koristi
- view = "featured" – ime pregleda
- Itemid = "435" – id stavke izbornika unutar tablice #__menu

Vrijednosti dobivene parsiranjem postavljaju se globalnu _GET varijablu i aktivira se onAfterRoute događaj:

```
JRequest::set($result, 'get', false);

// Trigger the onAfterRoute event.
JPluginHelper::importPlugin('system');
$this->triggerEvent('onAfterRoute');
```

Povratkom u JSite klasu provjerava se da li korisnik ima pravo pristupa stavki izbornika, u slučaju da nema preusmjerava ga se na formu za prijavu na sustav.

Sljedeće linije unutar index.php skripte kreiraju i pune traženu komponentu podacima pozivom dispatch metode.

```
// Dispatch the application.
$app->dispatch();
```

```
// Mark afterDispatch in the profiler.
JDEBUG ? $_PROFILER->mark('afterDispatch') : null;
```

Unutar ove metode kreira se dokument, odnosno JDocumentHtml, i postavljaju njegovi atributi. Dokument sadrži sve podatke o stranici te se ovaj objekt kasnije šalje u preglednik kako bi se stranica i prikazala.

U slučaju kada se radi o dokumentu koji je html tipa, sustav mu postavlja meta podatke, jezik koji se koristi unutar dokumenta i bazu stranice, odnosno URL do index.php skripte. Ako je riječ o dokumentu tipa feed (NewsFeed) postavlja se samo baza stranice. Također, bez obzira na tip, dokumentu se dodjeljuje naslovi i opis.

Dispatch metoda poziva i renderComponent funkciju koja se nalazi u sklopu JComponentHelper klasi.

```
$contents = JComponentHelper::renderComponent($component);
$document->setBuffer($contents, 'component');
```

Ova metoda dohvaća predložak, kojeg se dohvaća iz baze ako ne postoji u cache-u, i jezik koji se koristi unutar njega. Nakon što je predložak dohvaćen definiraju se konstante s putanjom do front-end i back-end datoteka komponente, a ovisno kojem dijelu aplikacije želimo pristupiti u varijablu $path se upisuje putanja do PHP datoteke komponente.

Nakon što sustav kreira putanju, dohvaća se jezik koji se koristi unutar komponente te se poziva metoda executeComponent kojom se izvršava kod komponente a rezultat, html kod, sprema u varijablu $contents.

```
// Execute the component.
$contents = self::executeComponent($path);
```

Tijelo ove metode dano je u nastavku:

```
protected static function executeComponent($path)
{
    ob_start();
    require_once $path;
    $contents = ob_get_contents();
    ob_end_clean();
    return $contents;
}
```

Metoda ob_start uključuje izlazni međuspremnik (engl. *output buffering*), odnosno sve što se vraća pokretanjem skripti unutar komponente ostaje zapisano u memoriji.

U nastavku je dan primjer učitavanja content komponente koju dohvaćamo koristeći putanju do content.php skripte unutar Components direktorija, a ova skripte sadrži sljedeće linije koda.

```
defined('_JEXEC') or die;

// Include dependancies
require_once JPATH_COMPONENT.'/helpers/route.php';
require_once JPATH_COMPONENT.'/helpers/query.php';

$controller = JControllerLegacy::getInstance('Content');
$controller->execute(JRequest::getCmd('task'));
$controller->redirect();
```

Ulaskom u skriptu dohvaćaju se ContentHelperRoute i ContentHelperQuery klase unutar helpers poddirektorija komponente. Kreira se objekt ContentController, upravljač komponente, za kojeg se definira zadani zadatak koji klasa izvršava, u ovom slučaju to je display metoda. Unutar konstruktora klase koju upravljač proširuje, JController, definira se putanja do direktorija modela i pregleda komponente.

Povratkom u skriptu datoteke izvršava se execute metoda unutar JController klase jer nije definirana u ContentController klasi.

```
// Make sure we have access
if ($this->authorise($doTask))
{
   $retval = $this->$doTask();
   return $retval;
}
```

Metoda provjerava da li korisnik ima pravo pristupa, a zatim poziva metodu čiji je naziv spremljen unutar $doTask varijable. U ovom slučaju poziva se ranije zadana metoda display.

Display metoda ContentController klase dohvaća naziv pregleda iz globalne varijable _REQUEST, a dohvaćena vrijednost je „featured“. Zatim se poziva istoimena metoda unutar JController klase, pomoću koje sustav kreira pregled i model komponente, odnosno ContentViewFeatured i ContentModelFeatured klasu, te dobiveni model dodaje unutar pregleda. U zadnjoj liniji ove metode poziva se display metoda pregleda.

Pregled poziva metode nad modelom, preko kojega dohvaća podatke i parametre koji se koriste za prikaz članaka unutar komponente, te ih obrađuje na temelju parametara i dodaje unutar klase pregleda.

Pozivom display metode JView klase koristi se metoda loadTemplate kako bi se učitao predložak komponente, a podaci postavili unutar njega.

Povratkom u content.php skriptu poziva se zadnja metoda nad upravljačem, redirect. Metoda služi za preusmjeravanje na određeni URL, a u slučaju da varijabla $redirect unutar upravljača nije postavljena, skripta nastavlja s daljnjim izvođenjem.

Povratkom u executeComponent metodu unutar $contents varijable sprema se html kod komponente pomoću funkcije ob_get_contents koja vraća sadržaj međuspremnika. Sadržaj u memoriji se zatim briše pomoću ob_end_clean metode.

Povratkom u dispatch metodu $contents varijabla se dodaje unutar dokument i aktivira onAfterDispatch događaj.

```
// Render the application.
$app->render();

// Mark afterRender in the profiler.
JDEBUG ? $_PROFILER->mark('afterRender') : null;
```

Unutar render metode dohvaća se dokument, korisnik i tip dokumenta. U ovom slučaju tip dokumenta je html, te se dohvaća predložak i njegovi parametri.

Nakon dohvaćanja podataka poziva se parse metoda unutar JDocumentHTML klase i prosljeđuju se dobiveni parametri.

```
public function parse($params = array())
{
   return $this->_fetchTemplate($params)->_parseTemplate();
}
```

Ova metoda prvo dohvaća predložak i jezik koji se koristi unutar njega koristeći _fetchTemplate metodu. Kod dohvaćanja predloška, kao i kod dohvaćanja komponente, koristi se izlazni međuspremnik u kojega se upisuje rezultat izvođenja index.php skripte predloška i upisuje u varijablu $contents.

Pomoću _parseTemplate metode u predlošku se traži regex izraz <jdoc:include\ type="([^"]+)" (.*)\/>, a rezultat se nakon obrade sprema u dokument unutar polja _template_tags.

Povratkom u render metodu aktivira se događaj onBeforeRender i poziva metoda render na dokumentu.

```
JResponse::setBody($document->render($caching, $params));
```

Unutar ove metode poziva se _renderTemplate metoda koja vrijednosti iz _template_tags polja zamjenjuje s HTML kodom, odnosno u ovom koraku se izvode skripte modula,

zaglavlja, poruka i debug elementa, a njihov rezultat se upisuje unutar predloška na mjesta jdoc:include linija. Kako rezultat izvođenja komponente već postoji, kopira se html kod komponente i postavlja na predviđeno mjesto u predlošku.

Dokument se zatim sprema unutar $body varijable JRespone klase. Prije povratka u index.php skriptu aktivira se onAfterRender događaj.

```
// Return the response.
echo $app;
```

Zadnja linija koda unutar index.php datoteke, echo naredba, poziva __toString metodu JApplication klase. Unutar ove metode provjerava se da li je potrebno napraviti kompresiju, a rezultat se prosljeđuje u JResponse toString metodu koja ovisno o proslijeđenom parametru kompresira stranicu, postavlja zaglavlja i šalje ih, te vraća varijablu sa sadržajem stranice koji se pomoću echo naredbe prikazuje u pregledniku.

Sustav završava izvođenje spremanjem sesije i zatvaranjem konekcije na bazu.

4.2 Joomla 3.x serija

Kao verzija za analizu Joomla 3.x sustava odabrana je posljednja verzija serije koju se moglo preuzeti za vrijeme izrade ovog rada, odnosno Joomla 3.9.4.

4.2.1 Instalacija

Instalacijski paket za Joomla 3.9.4 sustav preuzet je sa službenih stranica Joomla CMS.

Preuzeta je datoteka u zip formatu veličine 13,2MB te je zatim premještena i raspakirana na primarnom disku računala unutar WampServer direktorija. Točnije, u www direktoriju iz kojeg će se pokretati Joomla stranice. Nakon raspakiravanja veličina direktorija sustava iznosi 35,2MB.

Da bi započeli instalaciju, u pregledniku je potrebno upisati putanju do direktorija u kojem se nalazi Joomla, odnosno localhost/naziv_direktorija, nakon čega se otvara sučelje za instalaciju.

Kao što je prethodno navedeno proces instalacije Joomla 3.x sustava sastoji se od tri koraka.

U prvom koraku korisnik odabire jezik koji će se koristiti u procesu instalacije, te unosi osnovne podatke o sjedištu i administratorskom računu. Također u ovom koraku moguće je postaviti sjedište u neaktivno stanje, a ako pokušamo pristupiti neaktivnom sjedištu, u pregledniku se prikazuje obavijest da je stranica u izradi i forma za prijavu.

Pod osnovne podatke o sjedištu upisuje se naziv i opis sjedišta. Opis sadrži metapodatke koji se koriste od strane pretraživača.

Podaci vezani uz račun administratora uključuju e-mail adresu administratora, korisničko ime i lozinku za pristup administratorskom dijelu sustava.

U drugom koraku slijedi konfiguracija baze podataka. Potrebno je odabrati tip baze (MySQL, MySQLi ili MySQL (PDO)), unijeti ime servera na kojem se nalazi baza (najčešći naziv je localhost), korisničko ime i lozinku koji se koriste za pristup bazi, te ime baze podataka i prefiks naziva tablica unutar baze. Osim prije navedenih podataka postoji opcija za kreiranje sigurnosne kopije ili uklanjanje tablica iz prijašnjih instalacija sustava, uz uvjet da je prefiks tablice isti kao i onaj unesen u gore navedenoj formi.

U slučaju da baza nije prethodno kreirana, sustav prilikom instalacije sam kreira bazu i pri tome koristi podatke unesene u formu.

Zadnji korak ovog procesa služi za dovršetak instalacije. Stanica je podijeljena na dva dijela, u prvom je korisniku dana mogućnost instalacije uzorka jedne od gotovih sjedišta, a osim toga postoji opcija za slanje konfiguracijskih podatka na e-mail adresu unesenu u prvom koraku procesa instalacije. Ako se odabere slanje konfiguracijskih podataka, otvara se opcija slanja lozinke uz prije navedene podatke.

Drugi dio stranice sadrži četiri odjeljka u kojima su prikazani konfiguracijski podaci i postavke sustava.

Glavna konfiguracija sadrži informacije o sjedištu (ime, opis, status web sjedišta) i kreiranom administratorskom računu (e-mail adresa, korisničko ime, lozinka administratorskog računa).

Konfiguracija baze podataka prikazuje tip, korisničko ime, naziv baze podataka, prefiks naziva tablica baze i naziv servera na kojem se nalazi baza.

Prije instalacije sustava sve stavke predinstalacijske stavke provjere moraju biti zadovoljene, u suprotnom instalacije sustava neće biti moguća. Ove stavke su ustvari minimalni zahtjevi sustava koji moraju biti zadovoljeni.

Zadnji dio su preporučene postavke i odnose se na punu kompatibilnost PHP-a i sustava. Ako se postavke i razlikuju od preporučenih, to neće utjecati na nastavak instalacije ili pravilan rad sustava.

Pritiskom na gumb Instaliraj pokreće se instalacija, a ako je sustav pravilno instaliran prikazuje se stranica s porukom o uspješno završenom procesu. Ova stranica nudi i mogućnost instalacije dodatnih jezika, brisanje instalacijskog foldera, te gumbe za preusmjeravanje na administracijsku i front-end aplikaciju, odnosno web sjedište. Prije pristupa aplikacijama obavezno je ukloniti instalacijski direktorij.

4.2.2 Struktura direktorija sustava

Direktorij Joomla sustava sastoji se od:

- 6346 datoteka i 1967 direktorija prije instalacije
- 6105 datoteka i 1848 direktorija nakon instalacije

4.2.2.1 Cache direktorij

Kao i kod ranije opisanog Joomla 2.5 sustava ovaj direktorij koristi se kao privremena memorija sustava.

4.2.2.2 Components direktorij

Ovaj direktorij, kao i Joomla 2.5, sadrži sve komponente uključene u sustav, a svi direktoriji komponenti koriste isto pravilo pri imenovanju, odnosno ime direktorija započinju sa com_ prefiksom nakon čega dolazi naziv komponente. Razlika između verzija sustava je u broju komponenti koje one sadrže, tako Joomla 2.5 sadrži 11 komponenti, dok Joomla 3.9 dolazi s 18 uključenih komponenti na klijentskoj strani.

4.2.2.3 Includes direktorij

U ovom direktoriju se nalaze skripte za učitavanje konstanti i klasa potrebnih za pokretanje sustava, a u nastavku su opisane datoteke koje se nalaze unutar njega.

Defines datoteka sadrži definiciju globalnih varijabli koje sadrže putanje do direktorija koji se koriste unutar sustava.

Framework datoteka sadrži kod koji provjerava da li se u glavnom direktoriju sustava ne nalazi configuration.php datoteka i da li se unutar instalacijskog direktorija nalazi index.php. U slučaju da je ova provjera zadovoljena korisnika se preusmjerava na instalacijski proces te se prekida izvođenje ove skripte. Ako provjera nije zadovoljena, učitava se konfiguracija datoteka.

4.2.2.4 Language direktorij

Svi jezici instalirani unutar sustava nalaze se unutar ovog direktorija, a nakon instalacije ovdje se osim konstanti na engleskom jeziku nalaze svi jezici uključeni na stranici koja se prikazuje nakon uspješno obavljenog procesa instalacije. Moguće je instalirati jedan od 64 podržana jezika ili učitati jezični paket na isti način kao i u Joomla 2.5 sustavu.

4.2.2.5 Modules direktorij

Ovaj direktorij sadrži sve front-end module koji su instalirani na sustavu, a struktura direktorija i datoteka je ista kao i kod Joomla 2.5 sustava. Razlika je jedino u broju modula koji sustavi sadrže, tako Joomla 2.5 sadrži 24 modula, dok Joomla 3.9 sustav sadrži 25 modula na klijentskoj strani sustava.

4.2.2.6 Templates direktorij

Ovaj direktorij sadrži poddirektorije predložaka koji su instalirani unutar sustava, te su namijenjeni korištenju na klijentskoj aplikaciji. Struktura direktorija i datoteka unutar ovog direktorija je ista kao i kod Joomla 2.5 sustava uz iznimku da Joomla 3.x verzija uz CSS sadrži i LESS datoteke.

4.2.2.7 Administrator direktorij

Kao i u početnom direktoriju sustava, unutar administrator direktorija se nalaze cache, components, includes, language, modules i templates poddirektoriji koji su prethodno opisani. Osim navedenih poddirektorija, administracija sadrži help, logs i manifest direktorije.

Unutar help direktorij nalaze se helpsite.xml datoteka s poveznicom na Joomla dokumentaciju i poddirektorij koji sadrži toc.json datoteku s ključnim riječima, odnosno nazivom komponente ili značajke za koje postoji stranica s opisom i uputama za korištenje.

Unutar logs direktorija se spremaju informacije o aktivnosti sustava i njegovih proširenja, a primjer jedne od njih je zapisivanje grešaka koje su se dogodile tijekom izvođenja, te se one upisuju u error.php datoteku.

Manifest direktorij sadrži tri poddirektorija pod nazivima files, libraries i packages, a struktura ovog direktorija je ista kao i kod Joomla 2.5 sustava

4.2.2.8 Libraries

Ovaj direktorij sadrži biblioteke koje sustav koristi, a te biblioteke možemo podijeliti na vanjske koje su razvijene od treće strane i Joomla biblioteke.

Od vanjskih datoteka i programskih okvira ovaj direktorij sadrži:

- fof (engl. *framework on framework*) programski okvir koji se u sustavu koristi pri izradi proširenja, on olakšava i ubrzava ovaj proces te smanjuje količinu koda pri izradi
- idna_converter biblioteka služi za konverziju lokaliziranih (originalnih) naziva u internacionalizirane nazive domena
- phpass programski okvir se upotrebljava za hashiranje i provjeru lozinke
- php-encryption biblioteka se koristi za kriptiranje i dekripciju teksta
- phputf8 je biblioteka koja se koristi za obradu i manipulaciju UTF-8 teksta
- vendor sadrži pakete razvijene od treće strane i one Joomla klase čiji imenski prostor nije Joomla/CMS

Od Joomla biblioteke i programski okviri ovaj direktorij sadrži:

- cms sadrži Joomla biblioteke koje koriste sustav i programski okvir
- joomla sadrži Joomla programski okvir, odnosno Joomla platformu
- legacy sadrži klase koje su označene kao zastarjele (engl. *deprecated*), odnosno preporučuje se da se koriste nove klase kojima su obustavljene zamijenjene
- src sadrži klase važne za funkcioniranje Joomla sustav za upravljanje sadržajem definirane pod imenskim prostorom Joomla/CMS

Osim poddirektorija libraries sadrži i pet PHP datoteka:

- classmap – kreira aliase tako da poziva registerAlias metodu i prosljeđuje ime aliasa, putanju do klase i verziju sustava do koje će se alias koristiti
- cms – skripta kreira konstante putanje do platforme i konstantu verzije Joomla sustava, kreira autoloader, poziva classmap skriptu, učitava JLoader klasu i dodaje klase na autoload listu
- import.legacy – učitava klase unutar legacy direktorija
- import – skripta kreira konstante i učitava osnovne klase platforme
- loader – sadrži metode za učitavanje klase i obavljanje radnji nad aliasima

4.2.2.9 Installation direktorij

Unutar ovog direktoriju se nalaze cache, language i template direktorij koji su sadržani i u početnom Joomla direktoriju, a svrha i sadržaja ovih direktorija već je prethodno opisan.

Također, osim gore navedenih direktorija, unutar instalacijskog direktorija se nalaze i sljedeći poddirektoriji:

- application sadrži osnovne klase i skripte potrebne za učitavanje klasa i pokretanje aplikacije
- controller sadrži upravljača koji se koristi tijekom instalacije
- form sadrži dva poddirektorija, field poddirektorij sadrži klase za kreiranje i dohvaćanje odabrane vrijednosti polja instalacijskog jezika, prefiksa baze i uzorka podataka, dok se unutar rule poddirektorija nalaze klase koje se koriste za provjeru formata korisničkog imena i prefiksa baze
- helper sadrži database.php unutar koje se nalazi metoda za dohvaćanje JDatabaseDriver objekt
- html sadrži helper.php datoteku čija je zadaća u kreirati bočnu navigaciju i označiti trenutačan korak u procesu instalacije
- model sadrži modele koji se koriste prilikom instalacije
- response sadrži json.php datoteku unutar koje je definirana InstallationResponseJson klasa i njezin konstruktor, a koristi se za generiranje JSON-a unutar kojeg se spremaju poruke i greške koje su se dogodile tijekom instalacije
- sql sadrži SQL skripte za kreiranje i ažuriranje osnovnih tablica koje sustav koristi, kao i četiri skripte za punjenje baze s oglednim sadržajem
- view sadrži poglede koji su potrebni pri instalaciji, a svaki korak instalacijskog procesa ima svoj poddirektorij unutar kojeg se nalazi view.html.php datoteka s metodom za prikaz pogleda i tmpl direktorij s HTML kodom stranice

Osim direktorija unutar instalacijske aplikacije nalaze se i datoteke:

- index.php datoteku čijim se izvođenjem pokreće instalacijski proces sustava
- configuration.php-dist koja se koristi pri ručnoj instalacije sustava kada se ne koristi web preglednik
- COPYRIGHT sadrži tekst autorskih prava (GNU General Public License)
- INSTALL sadrži upute za instalaciju
- LICENSES sadrži popis licenci koje se koriste unutar sustava i njihov puni tekst
- localise.xml datoteka se koristi kako bi se postavio zadani jezik i ogledni podaci u procesu instalacije

4.2.2.10 Cli direktorij

Unutar cli (engl. *command-line-interface*) direktorija nalaze se aplikacije koje se izvršavaju pokretanjem pomoću komandne linije. Ovaj direktorij sadrži šest PHP skripti:

- deletefiles.php služi za uklanjanje zaostalih datoteka nakon ažuriranja sustava na višu verziju zato što postoji mogućnost da te datoteke prouzroče probleme u radu sustava
- finder_indexer.php služi za pokretanje Smart Search indeksacije
- garbagecron.php iz privremene memorije briše one podatke čije je vrijeme trajanja isteklo
- sessionGc.php briše podatke o sesijama koje su istekle
- sessionMetadataGc.php briše meta podatke isteklih sesija
- update_cron.php dohvaćanje informacija o ažuriranju za sva proširenja i pohranjuje ih u bazu podataka

4.2.2.11 Bin direktorij

Unutar ovog direktorija nalazi se keychain.php skripta koja pruža mogućnost sigurnog spremanja osjetljivih podataka, kao što su akreditacije kod pristupa, a prilikom enkripcije i dekripcije podataka koriste se javni i privatni ključevi.

4.2.2.12 Images direktorij

Unutar ovog direktorija, kao i kod Joomla 2.5 verzije sustava, nalaze se sve slike prikazane na web sjedištu.

4.2.2.13 Layouts direktorij

Unutar ovog direktorija se nalaze PHP skripte s HTML kodom koji sadrži strukturu za prikaz stavki na oba dijela aplikacije. Kako se neka stavka može koristiti na više mjesta, kao naprimjer u administracijskoj aplikaciji i front-end stranici, izlaskom Joomla 3.0 sustava sve stavke koje se prikazuju smještene su unutar ovog direktorija kako bi se pomoću JLayout klase mogle s jednog mjesta koristiti unutar više različitih pregleda (engl. *view*) i proširenja sustava. Također ako korisnik sustava poželi promijeniti strukturu ili izgled stavke u određenom predlošku potrebno je unutar direktorija predloška kreirati poddirektorij layouts i kreirati PHP skriptu s onim nazivom datoteke koji se koristi za prikaz stavke, odnosno nadjačati izvornu skriptu.

Ako u predlošku beez3 želimo nadjačati stavku koja definira izgled stavke u kojoj se ispisuje naziv autora članka, odnosno datoteka čija je putanja

\layouts\joomla\content\info_block\author.php, kreiramo datoteku s istim nazivom a putanja do te datoteke je \templates\beez3\layouts\joomla \content\info_block\author.php.

4.2.2.14 Media direktorij

Kao i kod Joomla 2.5 sustava unutar media direktorija nalaze se CSS i JavaScript datoteke koje se koriste unutar sustava, te su ovdje spremljene i slike, točnije ikone koje se prikazuju u sustavu. Jedina razlika između dvije verzije sustava je u tome što se kod Joomla 3.9 sustava unutar ovog direktorija mogu naći i LESS datoteke.

4.2.2.15 Plugins direktorij

Svi priključci Joomla CMS-a, kao i kod Joomle 2.5 sustava, proširuju klasu JPlugin. Pravilo pri imenovanju priključka kao i pojava XML datoteke te struktura njezinog sadržaja je isti u obje verzije sustava. Također Joomla 3.9 sadrži sve priključke kao i prije navedena verzija sustava.

Ovisno o svojoj zadaći i događajima koji ih aktiviraju priključci su svrstani u jedan od sljedećih direktorija, a za direktorije koji se nalaze i u prije analiziranoj verziji navedene su samo razlike.

Actionlog direktorij sadrži Joomla priključak koji bilježi promjene koje su se dogodile na sustavu. Pod promjene na sustavu smatra se svaka izmjena nad sadržajem, aplikaciji, proširenju, korisniku, grupi korisnika, itd.

Authentication direktorij, uz ranije navedene priključke, sadrži i autentifikacijski priključak cookie.

Captcha direktorij uz reCAPTCHA sadrži i nevidljivi (engl. *invisible*) reCAPTCHA koji od korisnika ne traži unos teksta, označavanje slika s određenim motivom ili označavanje polja s potvrdom da korisnik nije robot.

Fields direktorij sadrži priključke za stvaranje polja različitih tipova unutar proširenja koja podržavaju njihovo dodavanje.

Finder direktorij sadrži priključke koji služe za indeksiranje sadržaja za korištenje Smart Search-a. Indeksiranje je moguće provesti nad kategorijama, kontaktima, sadržajem, vijestima i tagovima.

Instaler direktorij sadrži priključke koji omogućavaju instalaciju paketa iz direktorija, URL-a i lokalno s korisnikova računala.

Privacy direktorij sadrži priključke koji dolaze unutar paketa za privatnost (engl. *Privacy Tool Suite*). Ovi priključci imaju zadaću prikazati korisniku formu i tekst za pristanak na pravila privatnosti stranice, te dohvatiti ili obrisati podatke o korisniku koji se nalaze unutar sustava.

Quickicon sadrži priključke koji sadrže metodu, onGetIcons, za provjeru određenih uvjeta. Ako ti uvjeti nisu zadovoljeni prikazuje se poruka upozorenja unutar administracijske aplikacije. Unutar ovog direktorija nalaze se:

- extensionupdate provjerava dostupnost ažuriranja za proširenja razvijena od treće strane i prikazuje obavijest ako je ažuriranje dostupno
- joomlaupdate provjerava dostupnost ažuriranja sustava i prikazuje obavijest ako je ažuriranje dostupno
- phpversioncheck provjerava verziju PHP-a instaliranog na sustavu korisnika i upozorava ako ta verzija nije u potpunosti podržana ili ne dobiva nove sigurnosne nadogradnja
- privacycheck provjerava da li postoje zahtjevi vezani za privatnost, kao na primjer kada korisnik zatraži svoje podatke, stariji od 14 dana i upozorava korisnika administracijskog dijela aplikacije na postojeće zahtjeve

Sampledata direktorij sadrži priključe koji se koriste uz ogledne stranice koje je moguće preuzeti u zadnjem koraku instalacije, odnosno nakon uspješne instalacije sustava.

Search direktorij sadrži priključke koji implementiraju dvije metode, onContentSearchAreas stvara niz stavki sadržaja koji će se pretraživati, dok onContentSearch metoda izvršava upit nad bazom, odnosno traži komponente. Jezgra Joomla sustava sadrži priključke za pretragu kategorija, kontakata, sadržaja, vijesti i tagovima.

Twofactorauth direktorij sadrži priključke za dvofaktorska autentifikaciju, a ovisno o izboru, korisnik može koristiti Google Authenticator totp ili YubiKey priključak.

4.2.2.16 Tmp direktorij

Ovaj direktorij koriste sustav i njegova proširenja kada je potrebno privremeno pohraniti neke podatke prilikom instalacije ili tijekom obrade datoteka.

4.2.2.17 Datoteke

Kao i kod Joomla 2.5 sustava direktoriji sadrže index.html datoteku, uz iznimku da kod Joomla 3.x sustava ovu datoteku sadrže samo oni poddirektoriji koji su hijerarhijski neposredno ispod glavnog direktorija.

Administrator direktorij sadrži index.php datoteku a samo dva njegova poddirektorija imaju index.html datoteku a to su cache i logs direktoriji.

Isto kao i administracijski, installation direktorij sadrži index.php skriptu a samo cache poddirektorij ima index.html datoteku.

Index.php skripta u ova dva direktorija, kao i u glavnom Joomla direktoriju, služi kao ulazna točka u aplikaciju.

Datoteke koje se nalaze unutar početnog Joomla direktorija su iste u obje analizirane verzije, uz manje izmjene samog sadržaja datoteci, te je i njihova namjena ista.

4.2.3 Baza podataka

Joomla koristi bazu podataka kako bi pohranila sve podatke koji se nalaze na sjedištu, a Joomla 3.9 verzija sustava podržava MySQL, Microsoft SQL server i PostgreSQL baze podataka.

Baza nakon instalacije sadrži 78 tablicu, a naziv svih tablica započinje prefiksom definiranim tijekom instalacije sustava. Unutar koda sve tablice koriste generički prefiks #__, u kojem se pri izvođenju upita u bazu znak # zamijeni prefiksom koji se nalazi u configuration.php, točnije s vrijednošću varijable $dbprefix.

Bazi podataka se može pristupiti preko programa phpMyAdmin koji omogućava pregled baze, kreiranje i brisanje tablica, manipulaciju i izvoz podataka koji se nalaze u bazi.

Tablice koje se koriste u jezgri Joomla sustava su:

- #__action_logs – unutar ove tablice bilježe se aktivnosti unutar sustava
- #__action_logs_extensions – sadrži komponente koje se prikazuju u konfiguraciji unutar opcije User Action Logs u administracijskoj aplikaciji
- #__action_logs_users – sadrži zapis u kojem se bilježi kojem korisniku i za koja proširenja će se na e-mail adresu slati obavijesti o aktivnostima korisnika
- #__action_log_config – sadrži komponente za koje se bilježe aktivnosti
- #__assets – sadrži popis svih komponenti, kategorija i članaka, te sadrži stupac u kojem je definirano koja grupa korisnika ima pravo vršiti određene radnje nad stavkom. Tablica se koristi svaki puta kada se provjerava da li je korisnik ovlašten za izvođenje određene radnje

- #__associations – koristi se u sustavima koji koriste više jezika da bi se određenu stavku (izbornika, naslov članka, naziv kategorije...) spojilo sa stavkom na drugom jeziku
- #__banners – popis svih bannera koji su kreirani unutar sustava
- #__banner_clients – podaci o klijentima bannera
- #__banner_tracks – podaci o praćenju bannera, odnosno njihovih pojavljivanja i klikova
- #__categories – popis svih kategorija koje su definirane unutar sustava
- #__contant_details – informacije o svim kontaktima definiranim na sustava
- #__content – članci kreirani unutar sustava
- #__contentitem_tag_map – mapira stavke sadržaj s tagovima koji su im dodijeljeni
- #__content_frontpage – poredak članaka unutar Featured Articles tipa stavke izbornika, aktivira se tako da se u Layout options pod opcijom Article Order odabere Featured Articles Order
- #__content_rating – sadrži prosječnu sumu i broj ocjena članka
- #__content_types – tipovi sadržaja definirani unutar sustava
- #__core_log_searches – popis pojmova pretraživanja unutar sjedišta
- #__extensions – popis proširenja instaliranih unutar sustava
- #__fields – popis polja (engl. *fields*) kreiranih unutar sustava
- #__fields_categories – mapira polja i pridruženu im kategoriju
- #__fields_groups – popis kreiranih grupa polja
- #__fields_values – sadrži vrijednosti unesene u polja
- #__finder_filters – filteri pretraživanja definirani u Smart Search-u
- #__finder_links – stavke sadržaja indeksirane pomoću Smart Search-a
- #__finder_links_terms – ovaj naziv koristi 15 tablica a razlikuju se po tome što su označene različitim sufiksima (znamenkama od 0 do 9 i slovima od a do f), mapira pojmove pretraživanja (#__finder_terms) i stavke iz prethodne tablice, te im dodjeljuje težinu
- #__finder_taxonomy – popis tipova stavki
- #__finder_taxonomy_map – mapira stavke sadržaja (#__finder_links) s tipovima iz prethodne tablice
- #__finder_terms – sadrži popis svih mogućih indeksiranih pojmova pretraživanja

- #__finder_terms_common – sadrži popis svih riječi koje se učestalo pojavljuju u jeziku, te koje se preskače tijekom indeksacije
- #__finder_tokens – privremena tablica koja se koristi tijekom indeksacije
- #__finder_tokens_aggregate – privremena tablica koja se koristi tijekom indeksacije
- #__finder_types – sadrži popis svih tipova komponenti
- #__languages – jezici instalirani unutar sustava
- #__menu – svi izbornici definirani unutar sustavu
- #__menu_types – popis svih izbornika definiranih na front-end stranici
- #__messages – sadrži sve poruke poslane od strane korisnika unutar sustava
- #__messages_cfg – bilježi promjene postavki u Private Messages Manager-u back-end korisnika
- #__modules – popis modula unutar sustava
- #__modules_menu – prikazuje povezanost modula i stavki izbornika, a pod menuid stupcem može sadržavati tri vrste vrijednosti: 0 označava da je modul povezan na sve stavke, pozitivan broj označava stavku na koju se modul veže, a negativan broj označava da se modul veže na sve stavke osim navedene
- #__newsfeeds – popis svih kreiranih stavki u Components -> Newsfeeds
- #__overrider – sadrži jezična nadjačanja koja se koriste na stranici
- #__postinstall_messages – sadrži popis naziva konstanti koje se nakon instalacije nude korisniku u administracijskom dijelu aplikacije
- #__privacy_consents – sadrži popis korisnika, tekst pravila privatnosti i datum pristanka korisnika
- #__privacy_requests – sadrži popis zahtjeva vezanih za privatnost
- #__redirect_links – sadrži kreirana preusmjeravanja sa starog na novi URL
- #__schemas – sadrži zapis za svako proširenje koje je napravilo izmjenu na bazi tijekom instalacije kao i verziju proširenja
- #__sessions – popis aktivnih sesija
- #__tags – sadrži tagove kreirane unutar sustava
- #__template_styles – predlošci instalirani na front-end i back-end strani sustava

- #__ucm[13]_base – sadrži ucm identifikator, identifikator tipa sadržaja iz #_categories tablice i identifikator samog sadržaja koji se dohvaća iz tablice unutar koje je taj sadržaja spremljen, svrha ove tablice je kreiranje jedinstvenog ključa sadržaja
- #__ucm_content – trenutno se koristi kako bi se određenom sadržaju dodijelio tag
- #__ucm_history – sadrži prijašnje verzije sadržaja i komponenti do kojih se može pristupiti preko opcije verzije (engl. *versions*) kada se odabere određena komponenta ili sadržaj unutar administracijske aplikacije
- #__updates – popis dostupnih paketa za instalaciju (ažuriranja)
- #__update_sites – popis poveznice na stranice s ažuriranjima
- #__update_sites_extensions – povezuje stavke iz tablice proširenja s poveznicama za ažuriranje
- #__usergroups – grupe korisnika definirane unutar sustava
- #__users – popis korisnika sustava
- #__user_keys – unutar tablice se sprema token kada korisnik unutar forme za prijavu označi opciju zapamti me
- #__user_notes – sadrži bilješke o korisnicima
- #__user_profiles – sadrži uparenje korisnika sa svakim poljem
- korisničkog profila koja se popunjavaju u User manageru
- #__user_usergroup_map - svaki red predstavlja korisnika i pridruženu mu grupu
- #__utf8_conversion – koristi se prilikom nadogradnje sustava s baze koja ne koristi utf8
- #__viewlevels - razine korisničkog prikaza

4.2.4 Tijek izvođenja

Kao i kod Joomla 2.5, ulazna točka ove verzije sustava je index.php datoteka. Na početku skripte definira se konstanta minimalne verzije PHP-a i provjerava da je verzija na kojoj se sustav pokreće veći ili jednak vrijednosti ove konstante.

```
/**
 * Define the application's minimum supported PHP version as a constant so
 * it can be referenced within the application.
 */
define('JOOMLA_MINIMUM_PHP', '5.3.10');
```

[13] UCM (engl. *Unified Content Model*) je pojam koji označava zajedničke karakteristike strukture sadržaja i komponenti. Cilj njegova uvođenja je smjestiti sadržaj i proširenja na jednom mjestu iz razloga što bi se time olakšalo kreiranje novih značajki i proširenja sustava jer se pri razvoju ne bi moralo paziti gdje je saržaj spremljen i kakve je strukture. UCM se prvi puta pojavljuje u Joomla! 3.1 verziji sustava

```
if (version_compare(PHP_VERSION, JOOMLA_MINIMUM_PHP, '<'))
{
   die('Your host needs to use PHP ' . JOOMLA_MINIMUM_PHP . ' or higher to  run this ve
}
```

Također se dohvaća vrijeme pokretanja i količina memorije koju sustav trenutno koristi, te se kao i u ranije analiziranoj verziji sustava definira __JEXEC konstantu kojoj se dodjeljuje vrijednost 1.

```
// Saves the start time and memory usage.
$startTime = microtime(1);
$startMem  = memory_get_usage();

/**
 * Constant that is checked in included files to prevent direct access.
 * define() is used in the installation folder rather than "const" to not
error for PHP 5.2 and lower
 */
define('_JEXEC', 1);
```

Sljedeće linije koda unutar skripte dohvaćaju defines.php skriptu koja sadrži putanje do određenih direktorija sustava.

```
if (file_exists(__DIR__ . '/defines.php'))
{
   include_once __DIR__ . '/defines.php';
}

if (!defined('_JDEFINES'))
{
   define('JPATH_BASE', __DIR__);
   require_once JPATH_BASE . '/includes/defines.php';
}
```

U prvom if bloku sustav provjerava da li se u korijenskom direktoriju nalazi defines.php datoteka, te ako ona postoji, sustav definira konstante unutar skripte. Putanju do trenutnog direktorija dobiva se pomoću konstante __DIR__, a vrijednost koju konstanta vraća jednaka je rezultatu metode dirname(__FILE__).

U sljedećem if bloku provjerava se da li je konstanta _JDEFINE definirana. Ako nije, sustav definira JPATH_BASE konstantu s putanjom do korijenskog direktorija sustava i poziva skriptu defines.php koja se nalazi unutar includes poddirektorija. Kod unutar ove datoteke dan je u nastavku.

```
defined('_JEXEC') or die;

// Global definitions
$parts = explode(DIRECTORY_SEPARATOR, JPATH_BASE);

// Defines.
define('JPATH_ROOT',          implode(DIRECTORY_SEPARATOR, $parts));
define('JPATH_SITE',          JPATH_ROOT);
```

```
define('JPATH_CONFIGURATION', JPATH_ROOT);
define('JPATH_ADMINISTRATOR', JPATH_ROOT . DIRECTORY_SEPARATOR .
'administrator');
define('JPATH_LIBRARIES',     JPATH_ROOT . DIRECTORY_SEPARATOR .
'libraries');
define('JPATH_PLUGINS',       JPATH_ROOT . DIRECTORY_SEPARATOR .
'plugins');
define('JPATH_INSTALLATION',  JPATH_ROOT . DIRECTORY_SEPARATOR .
'installation');
define('JPATH_THEMES',        JPATH_BASE . DIRECTORY_SEPARATOR .
'templates');
define('JPATH_CACHE',         JPATH_BASE . DIRECTORY_SEPARATOR . 'cache');
define('JPATH_MANIFESTS',     JPATH_ADMINISTRATOR . DIRECTORY_SEPARATOR .
'manifests');
```

Nakon što su konstante definirane sustav dohvaća framework.php skriptu koja se nalazi u includes poddirektoriju.

```
require_once JPATH_BASE . '/includes/framework.php';
```

Unutar ove skripte učitava se import.legacy.php skripta koja se nalazi u libraries poddirektoriju sustava.

```
// System includes
require_once JPATH_LIBRARIES . '/import.legacy.php';
```

Skripta definira JPATH_PLATFORM konstantu unutar koje se pohranjuje putanja do libraries direktorija, a zatim provjerava na kojem se OS-u sustav pokreće.

U sljedećem koraku provjerava se da li klasa JLoader postoji, a kako klasa još nije dohvaćena, koristi se naredba require_once i zatim provjerava da li je učitavanje klase prošlo uspješno.

```
// Import the library loader if necessary.
if (!class_exists('JLoader'))
{
   require_once JPATH_PLATFORM . '/loader.php';
}

// Make sure that the Joomla Loader has been successfully loaded.
if (!class_exists('JLoader'))
{
   throw new RuntimeException('Joomla Loader not loaded.');
}
```

Nakon učitavanja, poziva se setup metoda JLoader klase i prefiksu J unutar autoloader-a dodaje putanja do legacy poddirektorija (../naziv_projekta/libraries/legacy).

```
// Setup the autoloaders.
JLoader::setup();

JLoader::registerPrefix('J', JPATH_PLATFORM . '/legacy');
```

Setup metoda definira koje će se funkcije unutar JLoader klase koristiti za autoloader i definira prefiks J s putanjom do joomla poddirektorija (../naziv_projekta/libraries/joomla).

U ovoj verziji sustava, kao autoload funkcije koriste se load, _autoload, loadByPsr0, loadByPsr4 i loadByAlias metode te ih sustav poziva istim redom kako su ovdje navedene. Sadržaj metode setup dan je u nastavku.

```
public static function setup($enablePsr = true, $enablePrefixes = true,
$enableClasses = true)
{
    if ($enableClasses)
    {
        // Register the class map based autoloader.
        spl_autoload_register(array('JLoader', 'load'));
    }

    if ($enablePrefixes)
    {
        // Register the J prefix and base path for Joomla platform libraries.
        self::registerPrefix('J', JPATH_PLATFORM . '/joomla');

        // Register the prefix autoloader.
        spl_autoload_register(array('JLoader', '_autoload'));
    }

    if ($enablePsr)
    {
        // Register the PSR based autoloader.
        spl_autoload_register(array('JLoader', 'loadByPsr0'));
        spl_autoload_register(array('JLoader', 'loadByPsr4'));
        spl_autoload_register(array('JLoader', 'loadByAlias'));
    }
}
```

Povratkom u import.legacy.php skriptu sustav registrira, dodaje na listu autoloader-a, klase unutar JLoader objekta na način da u polje $classes dodaje element čiji je ključ naziv klase a vrijednost putanja do njezine datoteke.

```
// Register the PasswordHash lib
JLoader::register('PasswordHash', JPATH_PLATFORM .
'/phpass/PasswordHash.php');

// Register classes where the names have been changed to fit the autoloader
rules
// @deprecated  4.0
JLoader::register('JSimpleCrypt', JPATH_PLATFORM .
'/legacy/simplecrypt/simplecrypt.php');
JLoader::register('JTree', JPATH_PLATFORM . '/legacy/base/tree.php');
JLoader::register('JNode', JPATH_PLATFORM . '/legacy/base/node.php');
JLoader::register('JObserver', JPATH_PLATFORM .
'/legacy/base/observer.php');
JLoader::register('JObservable', JPATH_PLATFORM .
'/legacy/base/observable.php');
JLoader::register('LogException', JPATH_PLATFORM .
'/legacy/log/logexception.php');
JLoader::register('JXMLElement', JPATH_PLATFORM .
```

```
'/legacy/utilities/xmlelement.php');
JLoader::register('JCli', JPATH_PLATFORM . '/legacy/application/cli.php');
JLoader::register('JDaemon', JPATH_PLATFORM .
'/legacy/application/daemon.php');
JLoader::register('JApplication', JPATH_LIBRARIES .
'/legacy/application/application.php');
```

Povratkom u framework skriptu sustav dohvaća cms.php skriptu koja se nalazi u libraries direktoriju sustava.

```
// Bootstrap the CMS libraries.
require_once JPATH_LIBRARIES . '/cms.php';
```

Unutar ove skripte prefiksu J dodaje se putanja do cms direktorija (../naziv_projekta/libraries/cms).

```
// Register the library base path for CMS libraries.
JLoader::registerPrefix('J', JPATH_PLATFORM . '/cms', false, true);

/** @note This will be loaded through composer in Joomla 4 */
JLoader::registerNamespace('Joomla\\CMS', JPATH_PLATFORM . '/src', false,
false, 'psr4');
```

Sljedeća linija koda registrira imenski prostor (engl. *namespace*) autoloadera koristeći PSR-4 autoloading standard, te mu nadodaje putanju do src poddirektorija unutar libraries direktorija sustava.

Povratkom u cms.php skriptu dohvaća se autoloader.php datoteka unutar vendor poddirektorija koji je smješten u libraries direktoriju sustava. Pozivom ove skripte kreira se composer autoloader, te mu se dodjeljuju putanje do direktorija i datoteka unutar vendor poddirektorija koji se nalazi u libraries direktoriju sustava.

```
// Create the Composer autoloader
$loader = require JPATH_LIBRARIES . '/vendor/autoload.php';
$loader->unregister();
```

Sve gore navedene putanje definirane su unutar datoteke autoload_static.php koja je smještena unutar composer direktorija (../naziv_projekta/libraries/vendor/composer/autoload_static.php).

Povratkom u cms.php skriptu dodaje se metoda loadClass iz klase JClassLoader (../naziv_projekta/libraries/cms/class/loader.php) na popis metoda koje se koriste u autoloaderu. Ova metoda se dodaje na početak liste metoda autoloadera, iz razloga što je zadnji parametar metode postavljena na true, a to znači da će se ova metoda pozivati prva pri dohvaćanju klasa preko autoloadera.

```
// Decorate Composer autoloader
spl_autoload_register(array(new JClassLoader($loader), 'loadClass'), true,
true);

// Register the class aliases for Framework classes that have replaced
their Platform equivilents
require_once JPATH_LIBRARIES . '/classmap.php';
```

Dohvaćanjem classmap.php skripte sustav koristeći metodu registerAlias iz JLoader klase puni polje classAliases na način da je naziv klase s prefiksom J ključ, a vrijednost naziv imenskog prostora i klase na koju se veže.

Za primjer ćemo uzeti alias JRegistry i vrijednost \Joomla\Registry\Registry koja mu se pridružuje. Ako pogledamo datoteku na putanji ../naziv_projekta/libraries/vendor/joomla/registry/src/registry.php, vidimo da ona ima definiran imenski prostora kojem pripada (namespace Joomla\Registry) a na kojeg se nadodaje naziv klase.

Kako je proces autoloada klasa opširniji i kompliciraniji kod Joomla 3.x sustava, na primjeru klase JFactory dan je kratak opis koraka ovog procesa.

1. loadClass (ClassLoader klasa) traži klasu unutar polja classMap a kako klasa nije pronađena slijedi pretraživanje po prefiksu J unutar polja prefixLengthsPsr4 (te fallbackDirsPsr4) i prefixesPsr0 (te fallbackDirsPsr0). Kako klasa nije pronađena niti u jednom polju, naziv klase se upisuje u missingClasses polje. U suprotnom metoda bi vratila putanju do datoteke u kojoj se klasa nazivi i dohvatila je pomoću metode include. Također za dohvaćenu klasu kreirao bi se alias
2. load (JLoader klasa) traži klasu unutar classes polja te ju učitava pomoću putanje ako je klasa pronađena, kako se JFactory ne nalazi unutar ovog polja metoda vraća false kao rezultat izvođenja
3. _autoload (JLoader klasa) traži metodu unutar polja prefixes po prefiksa J i prolazi kroz sve putanje koje su navedene za taj prefiks. Kako se datoteka klase JFactory ne nalazi unutar ovih putanja (direktorija) sustav vraća false
4. loadByPsr0 (JLoader klasa) traži klasu unutar polja namespace (imenski prostor), odnosno unutar polja psr0 koje se nalazi u polju namespace. Kako ni jedan imenski prostor nije definiran unutar psr0 metoda vraća false
5. loadByPsr4 (JLoader klasa) traži klasu unutar polja namespace (imenski prostor), odnosno unutar polja psr4 koje se nalazi u polju namespace. (C:\wamp64\www\Joomla_394\libraries/src)

6. loadByAlias (JLoader klasa) traži klasu unutar polja classAliases. Unutar ovog polja sustav dohvaća imenski prostor klase (Joomla\CMS\Factory) i poziva metodu class_exists koja nas preusmjerava na loadClass metodu s početka ovog procesa. Povratkom u loadByPsr4 metodu sustav nalazi Joomla\CMS imenski prostor i dohvaća njegovu putanju (../naziv_projekta/libraries/src) na koju zatim dodaje naziv klase (zadnji element imenskog prostora dobivenog u metodi loadByAlias) i .php sufiks. Na kraju izvođenja metode klasa se dohvaća pomoću include_once naredbe kojoj prosljeđujemo putanju datoteke klase.

U nastavku cms.php na listu metoda autoloadera dodaje se autoload_fof_core metoda iz FOFAutoloaderFof klase, postavlja se metoda render iz klase JErrorPage za obradu grešaka za koje sustav nema definiran način hvatanja, a unutar JLoadera registriraju se klase za instalaciju proširenja i aliasi JAdministrator i JSite kojima se istim redom dodaju vrijednosti JApplicationAdministrator i JApplicationSite.

Povratkom u framework.php skriptu slijedi if blok u kojem se provjerava da li se radi o instalaciji ili pokretanju sustava. U slučaju da je riječ o instalaciji, tijek izvođenja se preusmjerava na instalacijski proces pomoću header metode koja šalje HTTP zaglavlje, odnosno lokaciju instalacijske index.php datoteke. U suprotnom, učitava se configuration.php datoteka, odnosno JConfig klasa i na temelju podataka iz konfiguracije postavlja se koje će greške sustav ispisivati.

Povratkom u index.php skriptu izvode se sljedeće linije koda.

```
// Set profiler start time and memory usage and mark afterLoad in the
profiler.
JDEBUG ? JProfiler::getInstance('Application')->setStart($startTime,
$startMem)->mark('afterLoad') : null;

// Instantiate the application.
$app = JFactory::getApplication('site');
```

Kako je JDEBUG konstanta postavljena na vrijednost 0 ova linija koda se preskače, odnosno rezultat izvođenja je null.

U drugoj liniji koda sustav dohvaća klijentsku aplikaciju sustava tako da poziva getApplication metodu s parametrom site unutar JFactory klase. Ova metoda zatim poziva getInstance metodu CMSApplication klase i prosljeđuje isti parametar.

```
self::$application = CMSApplication::getInstance($id);
```

Tijelo ove metode dano je u nastavku.

```
public static function getInstance($name = null)
{
    if (empty(static::$instances[$name]))
    {
        // Create a CMSApplication object.
        $classname = '\JApplication' . ucfirst($name);

        if (!class_exists($classname))
        {
            throw new
\RuntimeException(\JText::sprintf('JLIB_APPLICATION_ERROR_APPLICATION_LOAD'
, $name), 500);
        }

        static::$instances[$name] = new $classname;
    }

    return static::$instances[$name];
}
```

Kao što se vidi iz koda, sustav na \JApplication string dodaje prosljeđeni parametar Site i sprema ga u varijablu $classname koja se zatim koristi za kreiranje novog objekta, odnosno SiteApplication klase koja je definirana pod aliasom JApplicationSite.

Prilikom kreiranja novog SiteApplication objekta poziva se konstruktor klase, a naknadno se pozivaju i konstruktori klasa koje se međusobno proširuju, odnosno CMSApplication i WebApplication klase. CMSApplication klasa kreira objekt JEventDispatcher, zadužen za upravljanje događajima, i kreira ili dohvaća postojeću sesiju iz baze, dok se unutar konstruktora WebApplication klase kreiraju objekti (Input, Registry i WebClient) potrebni za izvođenje aplikacije, učitava konfiguracija i postavlja URI sustava.

Povratkom u index.php skriptu na $app varijabli, odnosno SiteApplication objektu, poziva se execute metoda.

```
// Execute the application.
$app->execute();
```

Metoda execute nalazi se u klasi CMSApplication i u prvoj liniji poziva metodu doExecute koja se nalazi unutar SiteApplication klase, metoda je prikazana u nastavku.

```
protected function doExecute()
{
    // Initialise the application
    $this->initialiseApp();

    // Mark afterInitialise in the profiler.
    JDEBUG ? $this->profiler->mark('afterInitialise') : null;

    // Route the application
    $this->route();

    // Mark afterRoute in the profiler.
```

```
    JDEBUG ? $this->profiler->mark('afterRoute') : null;

    /*
     * Check if the user is required to reset their password
     *
     * Before $this->route(); "option" and "view" can't be safely read
using:
     * $this->input->getCmd('option'); or $this->input->getCmd('view');
     * ex: due of the sef urls
     */
    $this->checkUserRequireReset('com_users', 'profile', 'edit',
'com_users/profile.save,com_users/profile.apply,com_users/user.logout');

    // Dispatch the application
    $this->dispatch();

    // Mark afterDispatch in the profiler.
    JDEBUG ? $this->profiler->mark('afterDispatch') : null;
}
```

Metode initialiseApp, route i dispatch su u svojoj suštini jednake kao i istoimene metode unutar index.php skripte Joomla 2.5 sustava, a metoda initialise je preimenovana u initialiseApp.

Metodom checkUserRequireReset sustav provjerava da li je od korisnika potrebno zatražiti promjenu lozinke, te je korisnika moguće preusmjeriti na stranicu njegova profila da bi izvršio promjenu.

Povratkom u execute metodu CMSApplication klase slijedi render metoda koja je kao i ranije navedene metode u suštini jednaka istoimenoj metodi koja se nalazi u index.php skripti Joomla 2.5 sustava. Ostatak metode execute dan je u nastavku.

```
// If we have an application document object, render it.
if ($this->document instanceof \JDocument)
{
    // Render the application output.
    $this->render();
}

// If gzip compression is enabled in configuration and the server is
compliant, compress the output.
if ($this->get('gzip') && !ini_get('zlib.output_compression') &&
ini_get('output_handler') !== 'ob_gzhandler')
{
    $this->compress();

    // Trigger the onAfterCompress event.
    $this->triggerEvent('onAfterCompress');
}

// Send the application response.
$this->respond();

// Trigger the onAfterRespond event.
$this->triggerEvent('onAfterRespond');
```

Unutar metode provjerava se i da li je u konfiguraciji sustava postavljeno kompresiranje stranice prije slanja u preglednik korisnika.

Metodom respond postavljaju se HTTP zaglavlja, koje sustav zatim i šalje, te se naredbom echo varijabla s HTML kodom stranice prikazuje u pregledniku.

```
$this->sendHeaders();

echo $this->getBody();
```

Sustav završava izvođenje spremanjem sesije i zatvaranjem konekcije na bazu.

4.3 Usporedba

Unutar ovog poglavlja dan je pregled glavnih razlika između analiziranih verzija sustava, odnosno Joomla 2.5.28 i Joomla 3.9.4 sustava, gdje se kao najbitnije razlike sa stajališta prosječnog korisnika može navesti temeljiti redizajn administratorskog sučelja kao i općeniti prelazak na responzivni dizajn unutar cijele aplikacije, što je postignuto ugradnjom Bootstrap biblioteke. Također, dolaskom 3.x serije sustav podržava i PostgreSQL baze podataka.

Iako je proces instalacije sa sedam koraka u Joomla 2.5 smanjen na tri koraka u Joomla 3.9 sustava, ova promjena je više kozmetičke naravi, odnosno korisnik i dalje unosi iste podatke u obje verzije sustava samo su ti podaci podijeljeni na manji broj stranica u Joomla 3.9 sustavu. Jedina realna promjena kod procesa je ta da je izbačen korak konfiguracije FTP-a, koji se u Joomla 2.5 sustavu nudio kao opcionalni.

Također treba navesti da razvojem 3.x serije sustav dobiva pregršt novih značajki i proširenja, kao što su tagovi, podrška za praćenje verzija sadržaja kao i Privacy Tool Suit proširenja nastala kao odgovor na GDPR direktivu Europske unije (tablice prikazuju svojstvena proširenja pojedine verzije).

Tablica 3 Proširenja svojstvena za Joomla! 2.5

Joomla! 2.5	
Naziv komponente	Opis
com_weblinks	Upravljanje vanjskim poveznica
Naziv modula	Opis
mod_weblinks	Formatirani prikaz vanjskih poveznica koje se nalaze unutar istoimene komponente

Naziv direktorija/priključak[14]	Opis
content/geshi	Prikazuje formatirani kod unutar članka (GeSHi sustav označavanja)
finder/weblinks	Koristi se kod indeksacije vanjskih poveznica (Smart Search)
quickicon/eosnotify	Prikaz upozorenja da instalirana verzija Joomle više nije podržana
search/weblinks	Omogućava pretraživanje vanjskih poveznica

Tablica 4 Proširenja svojstvena za Joomla! 3.9

Joomla 3.9	
Naziv komponente	Opis
com_actionlogs	Komponenta za rad s dnevnikom aktivnosti
com_ajax	Služi kao ulazna točka za HTTP zahtjev unutar modula, priključaka i predložaka, odnosno za dohvaćanje podataka unutar ovih proširenja
com_associations	Komponenta administratorske aplikacije (Components -> Multilingual Associations), koristi se kod sustava koji koriste višejezičnost kako bi neku stavku spojili s više prijevoda
com_contenthistory	Komponenta za rad s verzijama sadržaja
com_fields	Komponenta za kreiranje i prikaz polja (forme)
com_postinstall	Komponenta za prikaz poruka u administratorskoj aplikaciji nakon uspješne instalacije
com_privacy	Komponenta s pravilima korištenja i opcijama privatnosti na koje korisnik daje svoj pristanak, te za pregled zahtjeva predanih od strane korisnika a vezanih uz privatnost (uklanjanje ili izvoz podataka korisnika koji se nalaze unutar sustava)
com_tags	Komponenta za korištenje tagova
Naziv modula	Opis

[14] Naziv modula počinje prefiksom plg nakon kojega slijedi naziv direktorija u kojemu se priključak nalazi, te završava nazivom datoteke, npr. weblinks priključak unutar finder direktorija nosi naziv plgFinderWeblinks

mod_latestactions	U administratorskoj aplikaciji prikazuje listu zadnjih radnji korisnika (izmjene na sustavu, prijave i odjave sa sustava...)
mod_privacy_dashboard	Prikazuje tipove zahtjeve privatnosti korisnika (izvoz i uklanjanje podataka o korisniku) i njihov ukupan broj
mod_sampledata	Koristi se za instalaciju uzoraka podataka
mod_tags_popular	Prikazuje tagove koji se koriste unutar sjedišta
mod_tags_similar	Prikazuje poveznice na stavke sa sličnim tagovima
Naziv direktorij/priključak	Opis
actionlog/joomla	Bilježi radnje korisnika koje se vrše na proširenjima unutar sustava
authentication/cookie	Obrađuje kolačiće tijekom autentifikacije korisnika
captcha/recaptcha_invisible	Koristi Google invisible reCAPTCHA servis
content/confirmconsent	Dodaje tekst i odabirni okvir (engl. *checkbox*) u formu koja se prikazuje korisniku
content/contact	Kreira poveznicu između autora sadržaja i njegova profila
content/fields	Omogućava unos polja unutar uređivača tako da se naznači {field n}, gdje n označava id polja (engl. *field*)
editors-xtd/contact	Unutar uređivača sadržaja prikazuje gumb Contact
editors-xtd/fields	Unutar uređivača sadržaja prikazuje gumb Field
editors-xtd/menu	Unutar uređivača sadržaja prikazuje gumb Menu
editors-xtd/module	Unutar uređivača sadržaja prikazuje gumb Module
fields/	Ova skupina priključaka omogućava kreiranje polja različitih tipova
finder/tags	Koristi se kod indeksacije tagova (Smart Search)
installer/	Ova skupina priključaka omogućava instalaciju paketa s URL-a, lokalnog računala i iz direktorija
privacy/	Priključci vezani uz Privacy Tool Suite paket

quickicon/phpversioncheck	Provjerava instaliranu verziju PHP-a i vraća upozorenje ako instalirana verzija nije podržana
quickicon/privacycheck	Provjerava ako postoje neobrađeni zahtjevi privatnosti i prikazuje obavijest u administratorskoj aplikaciji
sampledata/	Dohvaća preuzete ogledne podatke
search/tags	Omogućava pretraživanje tagova
system/actionlogs	Bilježi radnje korisnika unutar sustava
system/fields	Prikazuje polja (engl. *fields*)
system/logrotation	Uklanja podatke iz loga nakon određenog broja dana (vrijednost 0 označava da se podaci ne brišu)
system/privacyconsent	Omogućava prikupljanje pristanak korisnika na uvjete korištenja i privatnosti
system/sessiongc	Čisti podatke isteklih sesija
system/stats	Koristi se za slanje podataka za statistiku na Joomla projekt server
system/updatenotification	Provjerava dostupnost novih nadogradnji sustava (nova verzija), ako pronađe novu verziju o tome e-mail porukom obavještava administratora sustava
twofactorauth/	Grupa priključaka za dvofaktorsku autentifikaciju (Google i YubiKey)
user/terms	Omogućava traženje pristanka korisnika na uvjete korištenja sjedišta

Ako pogledamo malo dublje u samu strukturu sustava vidljiv je utjecaj prilagodbe sustava s ciljem potpunog prelaska na PHP 7, što se najviše uočava kod uvođenja composer autoloadera (PSR standardi i classmap) i posljedično tome korištenjem imenskog prostora.

Uvođenjem imenskog prostora vidljive su promjene u strukturi direktorija, tako se većina klasa CMS-a u Joomla 3.9 sustavu nalazi unutar src poddirektoriju libraries direktorija, odnosno unutar imenskog prostora Joomla/CMS, dok su ostale Joomla klase koje spadaju u druge imenske prostore kao i klase razvijene od trećih strana smještene u vendor poddirektoriju.

Također, radi prilagodbe na korištenje imenskog prostora, datoteke unutar includes direktorija klijentske (application, menu, pathway i router datoteke) i administratorske (application, menu i router datoteke) aplikacije premještene su unutar Joomla/CMS imenskog prostora.

Od ostalih promjena unutar strukture, u Joomla 3.9 sustavu, na klijentskoj strani sustava dodani su direktoriji bin i layouts, dok je log direktorij premještena na administracijsku stranu. Također se uvodi i LESS, pa su stoga neke od CSS datoteka zamijenjene LESS-om.

Ako pogledamo tijek izvođenja, on je većinom ostao isti, uz iznimku da je proces inicijalizacije autoloadera proširen na korištenje composer autoloader, te je time i sam proces dohvaćanja klase nešto duži.

Unutar samog koda vidi se uvođenje novina predstavljenih u novijim verzijama PHP-a (5.3+), kao i refaktoriranje koda u kojima su određene metode preseljene u „roditeljske“ ili su kreirane potpuno nove klase.

Što se tiče usporedbi performansi sustava kao alat za mjerenje odabrana je Apache JMeter[15] aplikacija, a test je postavljen tako da se tijekom testiranja kreira 10 korisnika s razmakom od jedne sekunde, te svaki od korisnika sekvencijalno šalje HTTP zahtjeve za dohvat stranica redom kojim su navedene u popisu:

- Početna stranica klijentske aplikacije
- Članak Joomla!
- Članak Beginners
- Članak Professionals

Svaki od korisnika ponavlja zahtjev za dohvaćanje navedenih stranica 10 puta, što znači da se prilikom testiranja dohvaća ukupno 400 web stranica. Kao ogledni podaci korišteni su Default English Sample Data za Joomla 2.5 sustav i Learn Joomla English Sample Data u Joomla 3.9 sustavu, iz razloga što sadrže iste članke.

Tablica 5 Vremena izvođenja

	PHP	No caching	Conservative caching	Progressive caching
Joomla! 2.5	5.6.40	591 ms	406 ms	392 ms
Joomla! 3.9	5.6.40	924 ms	814 ms	796 ms
	7.3.1	226 ms	202 ms	196 ms

[15] Open source Java aplikacija za mjerenje performansi

Usporedbom performansi, odnosno vremena izvođenja analiziranih sustava, vidljivo je da se vrijeme izvođenja prepolovilo kod Joomla 3.9 sustava, u usporedbi s Joomla 2.5 sustavom, ali samo kada se koristi PHP 7.3.1.

Ako pogledamo slanje HTTP zahtjeva u Joomla 3.9 sustavu koji koristi PHP 5.6.40, vidimo da se u usporedbi s istim sustavom koji koristi PHP 7.3.1, latencija s prosječnih 800ms pada na 50ms.

5. Zaključak

Joomla sustav za upravljanje sadržajem u svojoj zadnjoj verziji, odnosno Joomla 3.x dobila je pregršt novih značajki i proširenja u odnosu na svojeg prethodnika. Većina promjena predstavljenih u 3.x seriji sustava proizišla je iz potrebe za implementacijom najnovijih trendova i rješavanja tehničkog duga, odnosno ostataka prethodno korištenih tehnologija.

Joomla 3.x seriju možemo promatrati kao pripremu, to jest tranzicijsku seriju, za Joomla 4 sustav, u kojoj vidimo prelazak na responzivni dizajn i postupnu pripremu sustava za PHP 7 čiji se pozitivni utjecaj već očituje u poboljšanju performansi sustava.

Sustav je oko sebe stvorio veliku zajednicu programera i dizajnera, koji značajno pridonose obogaćivanju sustava razvojem proširenja, te time i sustav čine dostupnijim većem broju korisnika zato što ovim proširenjima rješavaju probleme koje sama jezgra ne pokriva.

Također je važno spomenuti činjenicu kako je veliki broj web sjedišta baziran upravo na Joomla sustavu, te ga to čini privlačnom metom za napade od strane hakera, ali se ovaj problem rješava konstantnim (otprilike jednom na mjesečnoj bazi) izbacivanjem novih sigurnosnih zakrpa sustava, kao i detaljnom i lakodostupnom dokumentacijom pomoću koje se korisnik može dodatno zaštiti sljedeći preporuke Joomlinog tima zaduženog za sigurnost sustava.

Popis slika

Popis tablica

Literatura

Barker, D. (2016). *Web Content Management.* O'Reilly Media.

Boiko, B. (2004). *Content Management Bible 2nd edition.* Wiley.

Czysz, D. (16. srpanj 2014). *Joomla CMS Architecture*. Dohvaćeno iz SlideShare: https://pt.slideshare.net/dustinczysz/joomla-cms-architecture/5

Preston-Werner, T. (n.d.). *Semantic Versioning 2.0.0*. Dohvaćeno iz Semantic Versioning 2.0.0: https://semver.org/

Rouse, M. (Ožujak 2011). *Content management application (CMA)*. Dohvaćeno iz SearchContentManagement: https://searchcontentmanagement.techtarget.com/definition/content-management-application-CMA

Rouse, M. (Lipanj 2016). *Content management system (CMS)*. Dohvaćeno iz TechTarget: https://searchcontentmanagement.techtarget.com/definition/content-management-system-CMS

Rouse, M. (Srpanj 2017). *Content Management (CM)*. Dohvaćeno iz SearchContentManagement: https://searchcontentmanagement.techtarget.com/definition/content-management

Sharma, D. (n.d.). *Joomla Architecture: A brief Introduction.* Dohvaćeno iz Drexel University: http://www.pages.drexel.edu/~ds3222/eport/docs/joomla.pdf