PD.TVZ.HR
POLYTECHNIC & DESIGN Vol. 7, No. 4, 2019.
DOI: 10.19279/TVZ.PD.2019-7-4-01

RAZVOJ PRILAGOĐENOG PREDLOŠKA ZA WORDPRESS SUSTAV

RAZVOJ PRILAGOĐENOG PREDLOŠKA ZA WORDPRESS SUSTAV

Alen Šimec[1], Borna Grilec[2], Lidija Tepeš Golubić[1]

[1]Tehničko veleučilište u Zagrebu, Vrbik 8, 10000 Zagreb, Hrvatska
[2]Velečilište Velika Gorica, Zagrebačka ul. 5, 10410 Velika Gorica, Hrvatska

SAŽETAK

U ovom radu objasnili smo ideju i razvoj WordPress sustava za upravljanje sadržajem. Ukazali smo na prednost i mogućnosti korištenja prilagođenog predloška za specifično web rješenje. Na osnovu dva pristupa objasnili smo kako razviti prilagođeni predložak. Prikazali smo osnovne faze razvoja prilagođenog predloška klasičnim pristupom te modernim pristupom gdje se za razvoj predloška koristi WordPress REST API i Vue.js framework. Rubno smo u radu dotakli i na koji način dodaci mogu olakšati i poboljšati web stranicu.

Ključne riječi: *WordPress, prilagođen WordPress predložak, WordPress dodaci, decoupled WordPress, headless WordPress, WordPress REST API*

ABSTRACT

The authors have explained the idea and development of a WordPress content management system. The benefits and capabilities of using a custom template for a specific web solution have been highlighted. We have explained how to develop a custom template based on two approaches. The basic stages of developing a custom template with the classic approach and a modern approach have been displayed where the WordPress REST API and Vue.js framework are being used to develop the template. We have also mentioned how plugins can facilitate and enhance a website.

WordPress idea and development, custom WordPress template capabilities, WordPress plugins, WordPress template development, decoupled WordPress, headless WordPress, WordPress REST API

1. UVOD
1. INTRODUCTION

Danas se svako poduzeće i način poslovanja mora digitalizirati zbog konkurentnosti na tržištu te zbog toga nastaju web rješenja. Ključna je uloga interneta iako se u nekim situacijama nekvalitetnim i polovičnim rješenjima dešava da ulaganje u Internet poslovanje ne daje očekivane rezultate. Česta pogreška pojedinaca i poduzeća je u tome što traže izradu rješenja u brzom vremenskom roku.

Za slučajeve malih i srednjih moguća su kvalitetna rješenja korištenjem već gotovih sustava u kombinaciji s pažljivom prilagodbom rješenja. Isto tako, takvim rješenjima moguće je postići smanjenje vremena izrade rješenja, a da pritom kvaliteta ne strada, a na kraju se postiže optimalno rješenje za klijenta. Primjer takvog sustava je WordPress sustav za upravljanjem sadržajem čije gotovo beskonačne mogućnosti prilagodbe mogu poslužiti u svrhu izrade optimalnog rješenja u kraćem vremenskom periodu, na način da se samom klijentu omogući upravljanje sadržajem putem sučelja koje je prilagođeno korisniku (engl. user-friendly).

U prvom dijelu rada dajemo kratki pregled razvoja WordPressa i ključnih značajki WordPress sustava. Drugi dio obrađuje važnost izrade prilagođenih predložaka, njihove prednosti, mogućnosti i primjenu. Slijede faze izrade predloška za klasični i moderni pristup, te važnost korištenja dodataka (eng. plugins) u WordPress-u.

2. IDEJA I RAZVOJ WORDPRESS SUSTAVA

2. IDEA AND DEVELOPMENT OD WORDPRESS

Wordpress sustav stvoren je s ciljem da korisnicima olakša način objavljivanja sadržaja online. U lipnju 2001. godine Michael Valdrighi počeo je razvijati blog sustav pod nazivom b2 koji je bio PHP i MySQL alternativa tadašnjim poznatim platformama Blogger i Greymatter. Razvoj blog sustava b2 uspio je, no utvrđeni su i nedostaci. Zbog jednostavnosti je sustav bio poprilično zastupljen, čak se je stvorila i zajednica korisnika koji su konstantno davali povratne informacije kako bi se radilo na poboljšanju sustava.

Mike Little i Matt Mullenweg iskoristili su otvoreni kod b2 platforme i konstantno radili na poboljšanjima. Prva inačica WordPressa nastala je 1. travnja 2003. godine i objavio ju je Matt putem SourceForgea [1] Razvoj WordPressa traje i dalje, a konstantnim se unaprjeđenjima stvorila platforma čije su mogućnosti nadišle početnu ideju bloga.

WordPress je prerastao u sustav za upravljanje sadržajem (CMS-Content Management System) korišten na milijunima stranica i viđen od desetaka milijuna korisnika svakodnevno. WordPress je omogućio jednostavnost brzog objavljivanja, fleksibilnost u smislu kreiranja osobnog bloga, foto bloga, poslovne web stranice, profesionalnoga portfolija, medijskih portala, čak i web trgovina korištenjem dodataka (plugins). Neke od ključnih značajki WordPressa svakako su odlične mogućnosti objave sadržaja, optimizacija, upravljanje korisnicima te različitim ulogama korisnika, upravljanje medijskim zapisima, korištenje sustava za komentiranje i prijevod na više od 70 jezika kako bi se rad sa sustavom maksimalno pojednostavio svakome. Mogućnosti personalizacije WordPressa visoke su, tako je moguće koristiti već gotove teme (predloške), instalirati dodatke (plugins) koji jednim klikom mogu izmijeniti određene funkcionalnosti i web stranicu učiniti još boljom.

3. PREDNOST I MOGUĆNOSTI PRILAGOĐENOG PREDLOŠKA

3. ADVANTAGEA AND POSSIBILITIES OF ADJUSTED PLUGINS

WordPress omogućava korištenje već gotovih predložaka. Omogućen je pristup repozitoriju punom gotovih predložaka (besplatnih i onih s plaćanjem) koji mogu odgovarati potrebama određenog poduzeća, grupe ili pojedinca. Jedno od često postavljanih pitanja je, kako odlučiti odgovara li određenom korisniku jedan od gotovih ili besplatnih predložaka.[4] Ovdje je ključno upoznati brand koji će svoju priču ispričati online korisnicima. Ako korisnik želi ispričati svoju, jedinstvenu priču, velika je šansa da mu gotov predložak neće davati najbolje rezultate za definirane ciljeve. Kada govorimo o prilagođenom predlošku mnogi razmišljaju samo o izgledu predloška, a ne i o mogućnostima povezivanja određenog dijela predloška sa samim sustavom za upravljanje sadržajem. To znači da je određenu sekciju ili element prilagođenog predloška moguće uređivati i modificirati putem WordPress sučelja bez da administrator stranice mora ulaziti u programski kod. Zahvaljujući razvoju današnjih web tehnologija predložak možemo napraviti na više načina. Jedan način obuhvaća klasični pristup razvoju gdje se sav razvoj odvija unutar programskih datoteka WordPress-a, dok drugi način koji pruža značajne prednosti po pitanju korisničkog iskustva obuhvaća moderan pristup gdje se WordPress koristi samo kao administracijski dio, dok se za radni okvira koji nam omogućava korisnički pregled može koristiti neka druga tehnologija po izboru programera.

3.1. KLASIČNI PRISTUP RAZVOJU PREDLOŠKA

3.1. CLASSIC APROACH TO PLUGIN DEVELOPMENT

Klasični pristup razvoju prilagođenog predloška najčešći je i najstariji način razvoja. Većina gotovih predložaka koji se nalaze na repozitorijima razvijeni su prema tom principu.

Ovaj princip sav programski kod sadržava unutar WordPress sustava što je dobro zbog toga jer je na taj način moguće koristiti apsolutno sve funkcionalnosti WordPress sustava.

Ovim pristupom teško je razdvojiti vizualnog dijela za korisnike (engl. frontend) od administracijskog dijela (engl. backend) i najčešće se njihov razvoj provodi u isto vrijeme. Upravo zbog toga razvoj rješenja se može produljiti i samim time nastati loše organiziran kod. Također, ovaj pristup nema velike i najmodernije mogućnosti poboljšanja korisničkog iskustva.

3.2. MODERAN PRISTUP RAZVOJU PREDLOŠKA

3.2. MODERN APROACH TO PLUGIN DEVELOPMENT

Moderan pristup razvoju prilagođenog predloška interesantan je jer omogućava razdvajanje radnog okvira za korisnički pregled od administracijskog dijela.[4] To donosi prednosti po pitanju korisničkog iskustva i organizacije koda jer se na radnom okviru za korisnički pregled može koristiti npr. neki od modernih radnih okvira (engl. framework)za korisnički pregled kao što su Angular, React ili Vue, dok se u administracijskom dijelu koristi PHP na kojem je izrađena WordPress platforma. Također, ovim pristupom istovremeno je moguće razvijati administracijski dio i radni okvir za korisnički pregled što donosi znatno ubrzanje u razvoju. Za korištenje ovog pristupa potrebno je razumjeti REST API, JSON i JavaScript tehnologije što znači da je za ovaj princip razvoja potrebna viša razina znanja web tehnologija. Moderan pristup primjenjiv je otkako WordPress podržava REST API i na taj način već u temeljnoj verziji sadržava endpointe za CRUD postova, pristup korisnicima, kategorijama i različitim informacijama u obliku sirovih podataka. Mana ovog pristupa je u tome što na repozitoriju zasad postoji relativno mali broj dodataka (plugins) koji podržavaju REST API pa se jednostavnost proširivosti platforme otežava ako osoba koja razvija platformu nije educirana na području programiranja. Ovaj se pristup preporučuje u situacijama gdje je potrebno postići organizirani razvoj i visoko korisničko iskustvo.

4. FAZE RAZVOJA PRILAGOĐENOG PREDLOŠKA KLASIČNIM PRISTUPOM

4. PHASES OF DEVELOPMENT OF AJUSTED PLUGIN USING CLASSIC APROACH

Preuzmite WordPress CMS sa službene stranice https://wordpress.org/download/. Nakon toga je preuzeti WordPress potrebno raspakirati u javni folder unutar okruženja za razvoj, npr. XAMPP, MAMP ili neko drugo okruženje koje podržava PHP i MySQL. Kako bi se instalacija WordPress-a mogla izvesti uspješno, potrebno je još stvoriti praznu bazu podataka unutar phpmyadmin-a i pristupiti na rutu na kojoj se nalazi raspakirani WordPress za nastavak instalacije, u ovom slučaju to će biti „localhost/name-of-project“.

4.1. KREIRANJE PRILAGOĐENOG PREDLOŠKA

4.1. CREATING ADJUSTED PLUGIN

Kreiranje novog predloška vrši se unutar instalacije WordPressa na serveru. Potrebno je pozicionirati se u „wp-content/themes“ i tamo kreirati novu mapu proizvoljnog naziva predloška, u ovom slučaju zvati će se „classic-website“. Nakon toga slijedi stvaranje dva osnovna dokumenta, a to su style.css i index.php. Kod 1 prikazuje izgled style.css dokumenta. Svaki style.css dokument mora imati: definiran naziv teme (predloška), autora, opis predloška i verziju teme kako bi WordPress CMs mogao prepoznati predložak, a svi ostali detalji su proizvoljni.

```
/*
Theme Name: Classic Website
Author: My Name
Description: Classic Website
template
Version: 1.0
 */
```

Kod 1. *Neizostavni podaci na početku style.css dokumenta.*

Code 1. *Indispensable data at the beginning of the style.css document.*

4.2. KREIRANJE ZAGLAVLJA I PODNOŽJA
4.2 CREATING HEADER UND FOOTER

Budući da će više stranica koristiti isto zaglavlje i podnožje, WordPress omogućuje kreiranje zasebnih datoteka header.php i footer.php koje se po potrebi mogu pozivati na mjestima gdje je to potrebno. Kod 2 prikazuje izgled header.php datoteke koja se u dijelu stranice koja treba koristiti kod zaglavlja može pozvati pomoću već ugrađene funkcije get_header().

```
<!DOCTYPE html>
<html lang="hr">
    <head>
        <meta charset="utf-8">
        <title></title>
        <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
        <?php wp_head(); ?>
    </head>
    <body>
```

Kod 2. *Mogući izgled zaglavlja (header.php).*

Code 2. *Possible header layout (header.php).*

Osim zaglavlja, još jedan specifičan dokument koji se najčešće koristi više puta na web stranici je podnožje. Podnožje se sprema u footer.php dokument. Kod 3 prikazuje footer.php dokument koji se u dijelovima stranice koje ga zahtijevaju uključuje pomoću funkcije get_footer().

```
<footer>
    <p>Copyright - 2019</p>
</footer>
<?php wp_footer(); ?>
</body>
</html>
```

Kod 3. *Mogući izgled podnožja (footer.php).*

Code 3. *Possible footer layout (footer.php)..*

Važno je napomenuti kako trenutni izgled zaglavlja i podnožja nije finalni, već služi kao primjer što sve mogu sadržavati ovi dokumenti. S vremenom, kako se projekt razvija dodavat će se novi strukturni elementi.

4.3. ISPIS ŽELJENOG SADRŽAJA POMOĆU PETLJI (THE LOOP)
4.3. PRINT CONTENT USING LOOP COMMAND

Petlja (The Loop) u WordPressu određuje koji sadržaj prikazati na određenoj stranici na kojoj se nalazi krajnji korisnik. Sadržaj se može dohvaćati na temelju parametara koje petlja dohvaća iz URL-a ili na temelju prilagođenih, odnosno proizvoljnih parametara. WordPress „poznaje“ dvije vrste sadržaja, a to su postovi (posts) i stranice (pages) i njihovo dohvaćanje je moguće korištenjem petlje. Važno je napomenuti da se pojam „post“ u WordPressu može odnositi na bilo koju vrstu sadržaja, dakle postovi mogu biti: događaji, proizvodi ili bilo koja druga željena vrsta sadržaja.[3] Često se spominje kako je petlja (The Loop) „srce“ predloška/teme WordPressa budući da kontrolira na koji način se sadržaj prikazuje. Ona predstavlja funkcionalnu vezu između podataka MySQL baze podataka i izvedenog HTML koda koji se prikazuje krajnjem korisniku web stranice. Kako bi se pravilno prikazivao sadržaj petlju je potrebno smjestiti u odgovarajući template prema hijerarhiji koju definira WordPress i javno je dostupna na poveznici: https://developer.wordpress.org/themes/basics/template-hierarchy/. U slučaju kada se petlje postavljaju u datoteke s nazivima definiranim prema template hijerarhiji podaci će se dohvaćati prema parametrima iz URL-a, a sama petlja tada može izgledati kako prikazuje Kod 4. Prikazana petlja dohvatit će podatke ako postoje i omogućit će njihov ispis unutar određenog templatea.

```
<?php
 if ( have_posts() ) :
  while ( have_posts() ) : the_
post();
   //content to display (template
tags, HTML…)
  endwhile;
 endif;
?>
```

Kod 4. *Primjer osnovne petlje za prikaz sadržaja na temelju URL parametara.*

Code 4. *An example of a basic loop to display content based on URL parameters.*

Kada se ne bi slijedio princip template hijerarhije ili kada bi sadržaj petlje trebao ispisati negdje drugdje unutar programskih datoteka teme ili plugina WordPress-a tada je potrebno stvoriti petlju koja će dohvaćati podatke na temelju prilagođenih parametara, a ne na temelju URL-a. Jedina razlika je u tome što u tom slučaju prvo treba stvoriti polje argumenata, a zatim argumente predati objektu „WP_Query“ koja će omogućiti da se u petlju dohvati sadržaj prema prilagođenim zahtjevima te se npr. pomoću metode the_title() ispišu naslovi svih dostupnih postova. Primjer petlje na temelju prilagođenih parametara prikazuje Kod 5.

```
<?php
 $args = array(
  'posts_per_page' => '-1',
  'post_type' => 'post',
);
 $posts = new WP_Query( $args );
 while ( $posts -> have_posts() )
:  $posts -> the_post();
  //content to display
  the_title();
 endwhile;
 wp_reset_postdata();
?>
```

Kod 5. *Primjer složenije petlje za prikaz sadržaja na temelju prilagođenih parametara.*

Code 5. *An example of a more complex loop to display content based on custom parameters.*

Sada je moguće Kod 5 smjestiti u datoteku index.php i pristupom na početnu stranicu u ovom slučaju „localhost/classic-website“ ispisati će se naslovi svih dostupnih postova.

4.4. KORIŠTENJE FUNCTIONS.PHP DATOTEKE

4.4. USING THE FUNCTIONS.PHP FILE FROM WORDPRESS

U slučaju kada je potrebno definirati programski kod koji će uvijek biti na raspolaganju predlošku ili web stranici koristi se functions.php datoteka [2]. Unutar nje najčešće se sprema programski kod koji može uključivati različite stilove, skripte, navigacijske izbornike i slično. Kod 4 prikazuje functions.php datoteku i primjer uključivanja CSS datoteke koja će sadržavati stilove za web stranicu. Unutar funkcije default_styles() moguće je definirati stilove koje će koristiti web stranica, fontove, JavaScript datoteke te ostale skripte vezane uz sam predložak. Kako bi pridružene skripte mogle biti poznate WordPress-u potrebno ih je pridružiti pomoću funkcije add_action()i kuke (hook) „wp_enqueue_scripts“.

```
<?php
function default_scripts(){
  wp_register_style('style', get_
template_directory_uri() . '/style.
css', array(), '1.0');
  wp_enqueue_style('style');
}
add_action('wp_enqueue_scripts',
'default_scripts');
```

Kod 6. *Mogući izgled functions.php datoteke.*

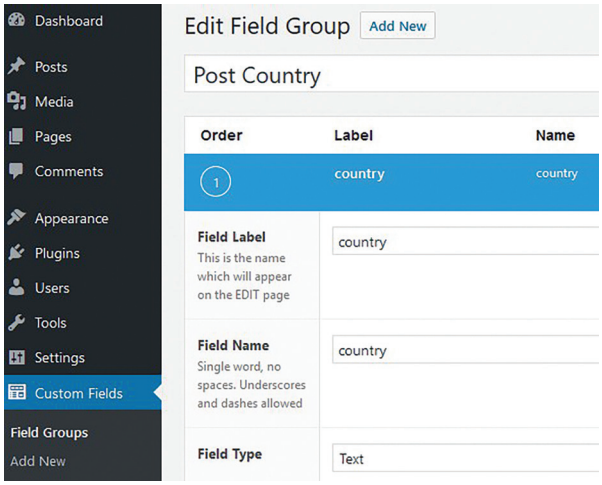
Code 6. *Possible layout of functions.php file*

4.5. INTEGRACIJA PRILAGOĐENIH POLJA U ADMINISTRATORSKO SUČELJE

4.5. INTEGRATION OF CUSTOM FIELDS INTO THE ADMIN INTERFACE

WordPress u temeljnoj izvedbi ne dopušta unos specifičnih podataka unutar administratorskog sučelja.[1] Kako bi se to omogućilo unutar admin sučelja potrebno je stvoriti nova polja za unos različitih tipova podataka. Jedan od načina je dodavanje novih „metabox“ elemenata programskim putem, dok je drugi način znatno lakši korištenjem dodatka (plugin) Advanced Custom Fields. Za početak je potrebno instalirati i aktivirati dodatak nakon čega je potrebno otvoriti karticu „Custom Fileds“ i dodati novu grupu. Slika 1 prikazuje polje koje će služiti za unos zemlje kojoj pripada određeni blog post.

Nakon definiranja novog polja potrebno je uči u sučelje za dodavanje novog posta i tamo će se vidjeti polje „country“ dodano pomoću ACF plugina.



Slika 1 *Sučelje za pretraživanje i instalaciju dodataka (plugins).*

Figure 1 *Plugin search and installation interface.*

Slika 2 *Izgled stvorenog polja unutar sučelja za dodavanje novih postova..*

Figure 2 *Layout of the created field inside the interface for adding new posts..*

Zadnji dio uključuje ispis sadržaja unutar petlje (The Loop) koja dohvaća sve postove. Tekstualni sadržaj moguće je dohvatiti pomoću funkcije the_field() kojoj se kao parametar predaje definirani naziv polja dodijeljen prilikom kreiranja polja u ovom slučaju „country“.

5. FAZE RAZVOJA PRILAGOĐENOG PREDLOŠKA MODERNIM PRISTUPOM

5. THE STAGES OF DEVELOPING A CUSTOM TEMPLATE WITH A MODERN APPROACH

Kako bi se prikazao postupak razvoja modernim pristupom koristit će se WordPress CMS za administracijski dio, dok će se za radni okvira koji nam omogućava korisnički pregled koristiti Vue.js framework. Ovaj princip izrade aplikacije pomoću WordPress-a i nekog od modernih radnih okvira za korisnički pregled frameworka često se naziva još i „decoupled“ ili „headless“ princip.

Postupak osposobljavanja okruženja i stvaranja WordPress projekta isti je kao što je to prikazano u poglavlju 4. Faze razvoja prilagođenog predloška klasičnim pristupom i obuhvaća postavljanje lokalnog okruženja za razvoj projekta, stvaranje baze podataka te skidanje, raspakiravanje i instalaciju WordPress-a. Ono što je dodatno potrebno učiniti je osposobiti okruženje za rad s Vue.js frameworkom. Instalacija Vue.js aplikacije izvršit će se korištenjem Vue CLI-a. Kako bi se mogao koristit Vue CLI, potrebno je instalirati Node.js kojega je moguće preuzeti na poveznici: https://nodejs.org/en/. Node.js služi kako bi se stvorilo serversko okruženje za rad s programima napisanim u JavaScriptu. Može se reći da je Node.js za JavaScript, ono što je XAMPP za PHP. Nakon što se preuzme Node.js i instalira se na računalo, potrebno je pokrenuti CMD naredbeni redak (Command Prompt).

Za instalaciju Vue CLI-a potrebno je unijeti naredbu npm i -g @vue/cli koja pomoću NPM-a instalira Vue CLI u okruženje. Budući da će se Vue.js koristiti za radni okvir za korisnički pregled, a WordPress za administracijski dio, sada se željena web stranica sastojati od dvije aplikacije od kojih je svaka zadužena za određenu svrhu.

5.1. RAZUMIJEVANJE I MOGUĆNOSTI WORDPRESS REST API-A

5.1. UNDERSTANDING AND OPTIONS OF WORDPRESS REST API

Pojam API (Application Programming Interface) predstavlja aplikacijsko programsko sučelje koje omogućava povezivanje dvije ili više različite vrste softvera i omogućava njihovu međusobnu komunikaciju.[2] API nije ništa novo i koristi se već dugi niz godina u svijetu razvoja softvera. Dobar primjer korištenja API-a su aplikacije koje koriste prijavu u svoje sustave korištenjem korisničkih računa kreiranih na društvenim mrežama (Facebook, Twitter, Google...). Uloga API-a je omogućavanje da programeri koriste sadržaj i značajke drugih aplikacija, servisa i platforma na siguran i ograničen način.

Kao i API, REST (Representational State Transfer) je također akronim i odnosi se na stil/arhitekturu izrade aplikacijskih programskih sučelja. Većina poznatih web servisa kao što su: Google, Facebook i Twitter koriste REST. REST je baziran na HTTP (Hyper Text Transfer Protocol) protokolu koji se na Internetu koristi za uspostavljanje konekcija i prijenos informacija. Prednost REST-a je u lakoći, fleksibilnosti i jednostavnosti rada s velikim količinama podataka i aktivnosti. Web servisi koji koriste REST paradigmu nazivaju se još i RESTful servisima. Princip rada REST-a temelji se na jednostavnosti, za razliku od složenih web servisa baziranih na SOAP-u ili XML-RPC-u. RESTful aplikacije uspostavljaju konekciju pomoću HTTP protokola, a korištenjem HTTP zahtjeva (HTTP request) obavljaju CRUD (Create, Read, Update, Delete) operacije nad podacima. Jedna od glavnih prednosti REST-a je mogućnost korištenja bilo kojeg programskog jezika za rad s podacima i potpuna neovisnost o platformi, npr. server može biti na Linux OS-u, a klijent na Windowsu. Po pitanju sigurnosti, RESTful aplikacije omogućavaju slanje korisničkih imena, lozinki i autentifikacijskih tokena unutar zaglavlja HTTP zahtjeva, a za potrebe enkripcije koristi se HTTPS (Hyper Text Transfer Protocol Secure) protokol što omogućava siguran prijenos informacija.

Još jedan pojam važan za shvaćanje REST API-a odnosi se na format izmjene podataka koji se najčešće izvodi pomoću JSON-a (JavaScript Object Notation). JSON je baziran na JavaScriptu i zanimljiv je zbog toga što je lako čitljiv strojevima i ljudima (machine and human-friendly). Upravo je JSON onaj koji omogućava platform-independent pristup, odnosno mogućnost rada s podacima korištenjem bilo kojeg programskog jezika jer većina jezika omogućava jednostavno interpretiranje JSON-a. Dakle, aplikacija A napravljena na jednom programskom jeziku može komunicirati s aplikacijom B izrađenom u drugom programskom jeziku tako da konvertira svoje podatkovne strukture u JSON ili suprotno.

WordPress REST API pomaže i omogućava izradu novih vrsta aplikacija korištenjem WordPressa tako da definira set funkcija pomoću kojih programeri mogu slati i primati zahtjeve putem Interneta.

Drugim riječima, WordPress REST API može vršiti interakciju sa servisima i web stranicama na Internetu koje mogu i ne moraju koristiti WordPress. Na taj način postiže se univerzalnost jer je interakcija i razmjena informacija moguća sa servisima koji mogu biti napravljeni u različitim programskim jezicima.

5.2. PRIPREMA WORDPRESS-A ZA REST API

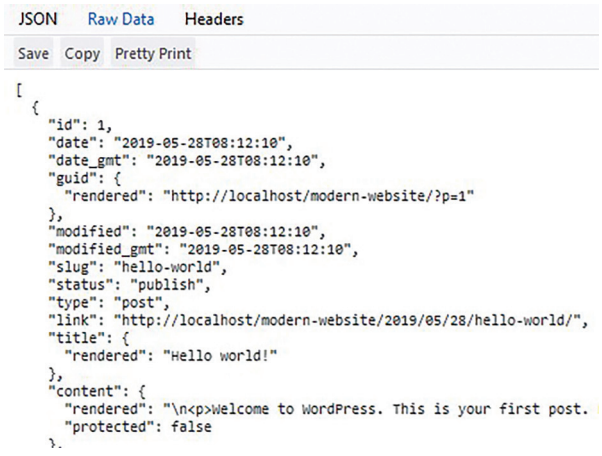
5.2. PREPARING WORDPRESS FOR THE REST API

Budući da na URL-u na kojem se nalazi WordPress nema potrebe prikazivati nikakav sadržaj, najbolje je kreirati prazan predložak. Kao i kod klasičnog predloška, potrebno je pozicionirati se u „wp-content/themes“ i dodati novu mapu s proizvoljnim nazivom teme npr. „modern-website“ i jedino što je potrebno kreirati unutar te mape su index.php i style.css datoteke. Datoteka index.php može ostati prazna, dok style.css mora imati tekst koji prikazuje Kod 7.

```
/*
Theme Name: Modern Website
Author: My Name
Description: Modern Website
template
Version: 1.0
 */
```

Kod 7. *Osnovni kod za novi predložak*

Code 7. *The basic code for the new template.*



Slika 3 *Prikaz podataka o postovima u JSON obliku.*

Figure 3 *View post data in JSON format.*

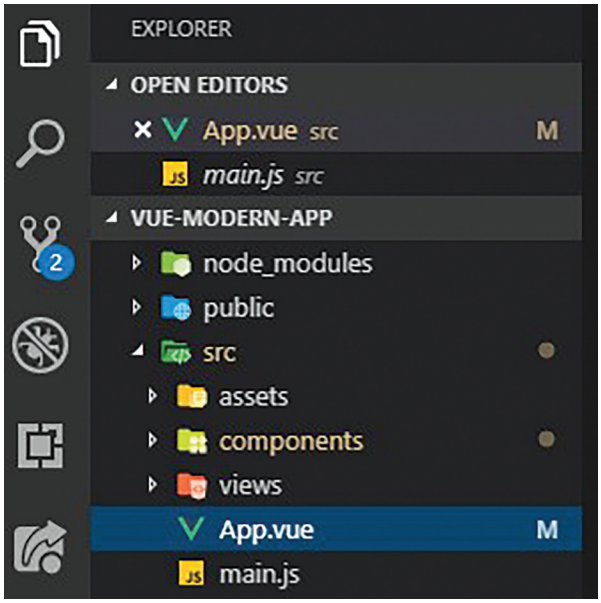
Sada ako se pristupi početnoj ruti na kojoj se nalazi WordPress dobit će se prazna stranica. Ono čemu želimo imati pristup to su sirovi podaci u JSON obliku i njima je moguće pristupiti dolaskom na početnu rutu i dodavanjem „/wp-json“, na taj način prikazuju nam se sve dostupne rute. Dolaskom na rutu npr. „localhost/modern-website/wp-json/wp/v2/posts“ prikazuju nam se podaci o svim postovima kako je prikazano na Slici 3.

5.3. PRIPREMA VUE.JS FRAMEWORKA I INTEGRACIJA SA WORDPRESS REST API-EM

5.3. VUE.JS FRAMEWORK PREPARATION AND INTEGRATION WITH WORDPRESS REST API

Kada je WordPress spreman, potrebno je osposobiti Vue.js aplikaciju koja će moći dohvaćati podatke sa WordPress REST API-a. Važno je napomenuti da za „decoupled“ princip razvoja nije nužno koristiti Vue.js, već se mogu koristiti i drugi okviri kao što su Angular, React itd., no za potrebe ovog rada koristit će se Vue.js. Kako bi se stvorila Vue aplikacija potrebno je u CMD upisati naredbu „vue create“ i željeni nayiv aplikacije, u ovom slučaju „vue create vue-modern-app“.[7] Nakon toga potrebno je slijediti postupak instalacije i po završetku se pozicionirati u direktorij instalacije pomoću naredbe „cd vue-modern-app“ te pokrenuti aplikaciju pomoću naredbe „npm run serve“ nakon čega će aplikacija biti dostupna na URL-u http://localhost:8080. Sada je potrebno otvoriti root folder Vue aplikacije i pozicionirati se datoteku App.vue. Folder Vue aplikacije izgleda kako je prikazano na Slici 4. Vue aplikacija sadržava puno različitih datoteka, no za potrebe ovog rada prikazati će se kako podatke možemo dohvatiti i prikazati unutar App.vue dokumenta koji se prikazuje kada se pristupi URL-u: http://localhost:8080.

Kako bi se podaci sa WordPress REST API-a mogli dohvatiti unutar Vue.js radnog okvira za korisnički pregled koristit će se Axios biblioteka koja omogućava pojednostavljeni rad sa HTTP zahtjevima.[5]



Slika 4 *Izgled datoteka Vue.js aplikacije.*

Figure 4 *Appearance of Vue.js application files.*

Za korištenje Axiosa potrebno je instalirati novu biblioteku unutar Vue aplikacije. Instalacija Axios biblioteke izvršit će se pomoću NPM-a (Node Package Manager). Unutar naredbenog retka potrebno je upisati npm install axios nakon čega se biblioteka pojavljuje u direktoriju „node_modules“.[3] Za njeno korištenje potrebno ju je uključiti u datoteku koja će koristiti mogućnosti Axiosa i to je moguće učiniti pomoću import deklaracije. Sada kada je sve spremno unutar datoteke App.vue potrebno je unutar script tagova uključiti axios biblioteku, stvoriti prazno polje za postove „posts“ i u mounted() kuki (hook) dohvatiti podatke o postovima sa WordPress-a u polje „posts“. Ovaj princip prikazan je na Kodu 8.

```
<script>
import axios from "axios";
export default {
  data() {
    return {
      posts: []
    };
  },
  mounted() {
    axios
      .get("http://localhost/
modern-website/wp-json/wp/v2/
posts")
```

```
        .then(response => {
          this.posts = response.
data;
        });
    }
};
</script>
```

Kod 8. *Osnovni kod za novi predložak*

Code 8. *Retrieving data from WordPress into App.vue.*

Dohvaćeni se podaci nalaze u polju „posts“ i lako ih je ispisati unutar App.vue komponente u dijelu između template tagova pomoću v-for direktive. Kod 9 prikazuje mogući princip ispisa svih WordPress postova sa naslovima i sadržajem.

```
<template>
  <div id="app">
    <div v-for="post in posts"
:key="post.id">
      <h2 v-html="post.title.
rendered"></h2>
      <p v-html="post.content.
rendered"></p>
    </div>
  </div>
</template>
```

Kod 9. *Ispis podataka o postovima u App.vue.*

Code 9. *Print post data in App.vue.*

Zahvaljujući ovom principu podaci WordPress-a mogu se dohvatiti u Vue aplikaciju koja svojim modernim konceptima izrade radnog okvira za korisnički pregled može znatno pospješiti korisničko iskustvo i organizaciju razvoja WordPress web stranice. Naravno u ovom radu prikazano je kako se mogu dohvatiti podaci na najjednostavniji način, no u praksi se primjenjuju inteligentniji principi za upravljanje podacima koji su specifični za Vue framework. Iako se cijela priča sada čini kompleksnija od klasičnog pristupa razvoju WordPress predloška, ovaj princip omogućava reaktivno vezanje podataka, MVVM arhitekturu, upravljanje životnim ciklusom aplikacije, komponentni dizajn i još mnoge druge mogućnosti koje nudi Vue.js i moguće ih je saznati u službenoj dokumentaciji frameworka na poveznici: https://vuejs.org/v2/guide/. Ovakvim pristupom većina funkcionalnosti WordPress-a se zadržava, isto tako ono što je najvažnije, administratorsko sučelje WordPress-a ostaje isto i sav sadržaj može se dodavati i upravljati preko tog sučelja, jedina je razlika u tome što se podaci prikazuju pomoću Vue aplikacije koja komunicira sa WordPress-om. Za ovaj pristup potrebna je viša razina znanja JavaScripta nego za klasični pristup razvoju predloška.[2] Također je važno spomenuti da je unutar sučelja WordPressa sada isto moguće integrirati dodatna polja kao što se je to radilo u poglavlju 4.5. Integracija prilagođenih polja u administratorsko sučelje, samo je potrebno koristiti dodatak (plugin) „ACF to REST API“ koji integrirana polja prikazuje na REST rutama.

6. VAŽNOST KORIŠTENJA DODATAKA (PLUGINS)

6 THE IMPORTANCE OF USING PLUGINS

Prilagodba web rješenja veže se za korištenje dodataka (plugins). Dodaci zapravo predstavljaju dodatne komponente koje se mogu implementirati u određeno web rješenje. Te dodatne komponente su programski kodovi koje pišu programeri za neke svoje specifične aplikacije ili slučajeve. [1]

Većina dodataka nije prilagođena za REST API te će u slučaju razvoja predloška modernim pristupom biti teže pronaći plugin koji će odgovarati određenoj svrsi. U slučaju klasičnog pristupa razvoju predloška izbor dodataka je velik i WordPress CMS biti će lako proširiti novim funkcionalnostima.

Neki od neizostavnih dodataka svakako su dodaci za optimiziranje sadržaja za tražilice, gotovi kontakt obrasci te dodaci za kreiranje prilagođenih polja unutar sučelja kako bi se ostvarilo lako povezivanje s predloškom. [4]

7. ZAKLJUČAK

7. CONCLUSION

WordPress omogućava visoku razinu prilagodbe web rješenja. Glavne prednosti koje smo u ovom radu zaključili su jednostavnost korištenja sučelja od strane administratora Interent stranice CMS sustava i brzina razvoja Internet rješenja od strane programera.

Zahvaljujući protokolu REST API-u i modernom pristupu razvoja prilagođenog predloška mi smo zaključili da je moguće dodatno poboljšati korisničko iskustvo i performanse WordPress Internet stranica, a da se administratorsko sučelje zadrži. Što se tiče modernog pristupa razvoju predloška, on najviše ovisi o poznavanju Vue.js frameworka ili nekog drugog radnog okvira za korisnički pregled, jer je u tom slučaju važno samo da se podaci sa WordPress REST API-a dohvate na radni okvira za korisnički pregled, a što će se dalje raditi ovisi o razvojnom timu. Mi smatramo da rad s WordPress CMS sustavom bez koda koji je nepotreban i ugrađivanjem protokola koji šalje podatke radnom okviru Vue.js podrazumijeva jedinstveno i kvalitetnije rješenje u razvoju i konačnoj izvedbi aplikacije.

8. REFERENCE

8. REFERENCES

[1.] Milestones: The Story of WordPress; str. 34; Wordpress.

[2.] Williams, B; Damastra, D; Stern, H: Proffesional WordPress: Design and Development, Second Edition, John Wiley & Sons, 2013.; ISBN-13: 978-0470560549

[3.] Šimec A.; Nožica B.; Virtualizacija uredskog poslovanja; Tiskarstvo & Dizajn 2014.; ISBN: 9789537064235

[4.] Filipova Olga; Learning Vue.js 2; Packt Publishing, 2016.; ISBN: 9781786461131

[5.] Šimec Alen, Davor Lozić; Extending PHP with modules; Polytechnic & Design; Vol. 3, No. 1, 2015.; ISSN: 1849 – 1995

[6.] Djirdeh, H; Murray, N; Lerner, A: Fullstack Vue: The Complete Guide to Vue.js; Fullstack.io, 2018.; ISBN-13: 978-1987595291

[7.] Šimec, Alen; Lozić, Davor; Basic PHP Implementations, Opcodes and Internal Work; MIPRO 2015 - 38. međunarodni skup; ISSN: 1847-3946.

[8.] Šimec, Alen; Golubić Tepeš Lidija; Analysis of Developers Choices for API Rest or Soap Protocols; International Journal of Circuits, Systems and Signal Processing; ISSN: 1998-4464; Volume 11, 2017

AUTORI · *AUTHORS*

● **Alen Šimec -** nepromjenjena biografija nalazi se u časopisu Polytechnic & Design Vol. 5, No. 2, 2017.

Korespondencija · *Correspondence*
alen@tvz.hr

● **Borna Grilec**
Završio studij održavanja računalnih sustava na VVG-u, a potom i studij informacijskih sustava, također na VVG-u. Tijekom studija imao sam izrazit interes za predmete vezane uz programiranje, poslovanje, management i komunikacije te sam već tijekom studija krenuo s freelancanjem na području web developmenta. Proteklih nekoliko godina radio sam s različitim tehnologijama kao što su: PHP, Java, JavaScript, HTML5, Sass, Vue.js, WordPress, C# itd. Isto tako redovito polazim stručne konferencije, workshope i meetupe te se primarno bavim frontend i wordpress developmentom. U zadnje vrijeme dosta se bavim i savjetovanjem startupa u njihovom daljnjem razvoju. Zadnjih godinu dana vodim vlastitu web agenciju koja odrađuje poslove iz područja dizajna, programiranja i digitalnog marketinga sa svrhom izrade prilagođenih web rješenja.

Korespondencija · *Correspondence*
bornagrilec@gmail.com

● **Lidija Tepeš Golubić -** nepromjenjena biografija nalazi se u časopisu Polytechnic & Design Vol. 5, No. 2, 2017.

Korespondencija · *Correspondence*
ltepes2@tvz.hr