Contents]

[Fatah Gabrial „WordPress Course - Beginners Guide to WordPress 5 (2022)” [Udemy] (2022) 2](#_Toc132534787)

[Brad Schiff „Become a WordPress Developer.. Unlocking Power With Code” [Udemy] (2022) 4](#_Toc132534788)

[PHP 6](#_Toc132534789)

[Teme (Themes) 7](#_Toc132534790)

[Postări (Posts) 9](#_Toc132534791)

[Antete şi subsoluri (Headers and Footers) 9](#_Toc132534792)

[Fişierul functions.php 9](#_Toc132534793)

[Pagini (Pages) 9](#_Toc132534794)

[Recomandări 10](#_Toc132534795)

[Meniuri de navigare dinamice 10](#_Toc132534796)

[Aspectul de blog al site-ului web 11](#_Toc132534797)

[Ecrane de arhive 11](#_Toc132534798)

[Interogări personalizate 12](#_Toc132534799)

[Tipuri de postări personalizate 12](#_Toc132534800)

[Câmpuri personalizate 13](#_Toc132534801)

[Ordonarea interogărilor personalizate – Sortarea 14](#_Toc132534802)

[Editarea interogărilor implicite 14](#_Toc132534803)

[Crearea unui şablon dedicat pentru o singură pagină 15](#_Toc132534804)

[Crearea de relaţii între conţinuturi ale diferitelor tipuri de postări 15](#_Toc132534805)

[Tipul de postare Professor 17](#_Toc132534806)

[Imagini caracteristice 17](#_Toc132534807)

[Banner de pagină dinamic 19](#_Toc132534808)

[Reducerea codului duplicat – funcţia get\_template\_part() 19](#_Toc132534809)

[Prepararea calculatorului şi proiectului pentru JavaScript 19](#_Toc132534810)

[Google Map 20](#_Toc132534811)

# Fatah Gabrial „WordPress Course - Beginners Guide to WordPress 5 (2022)” [Udemy] (2022)

* Descărcare WordPress:

<https://wordpress.org> => Get WordPress

* wordpress-6.0.2.zip - Size: 21,7 MB (22.772.803 bytes)
* Dezarhivare în folder-ul E:\work\vhosts\www.wordpress.bv\public\_html.
* Crearea bazei de date “blog” în MySQL.
* Instalare WordPress local:

[www.wordpress.bv](http://www.wordpress.bv)

* <http://www.wordpress.bv/wp-admin/setup-config.php>:
  + English (United States)
  + **wp-config.php**
    - Database Name => blog
    - Username => root
    - Password => 67ho48si14me,./
    - Database Host => localhost
    - Table Prefix => wp\_

Submit

* + Run the installation
  + Information needed:
    - Site Title => Blog
    - Username => admin
    - Password => wVw1C!f1C8glwxamE9
    - Your Email => mariustamba22@gmail.com
    - Search engine visibility => Neselectat

Install WordPress

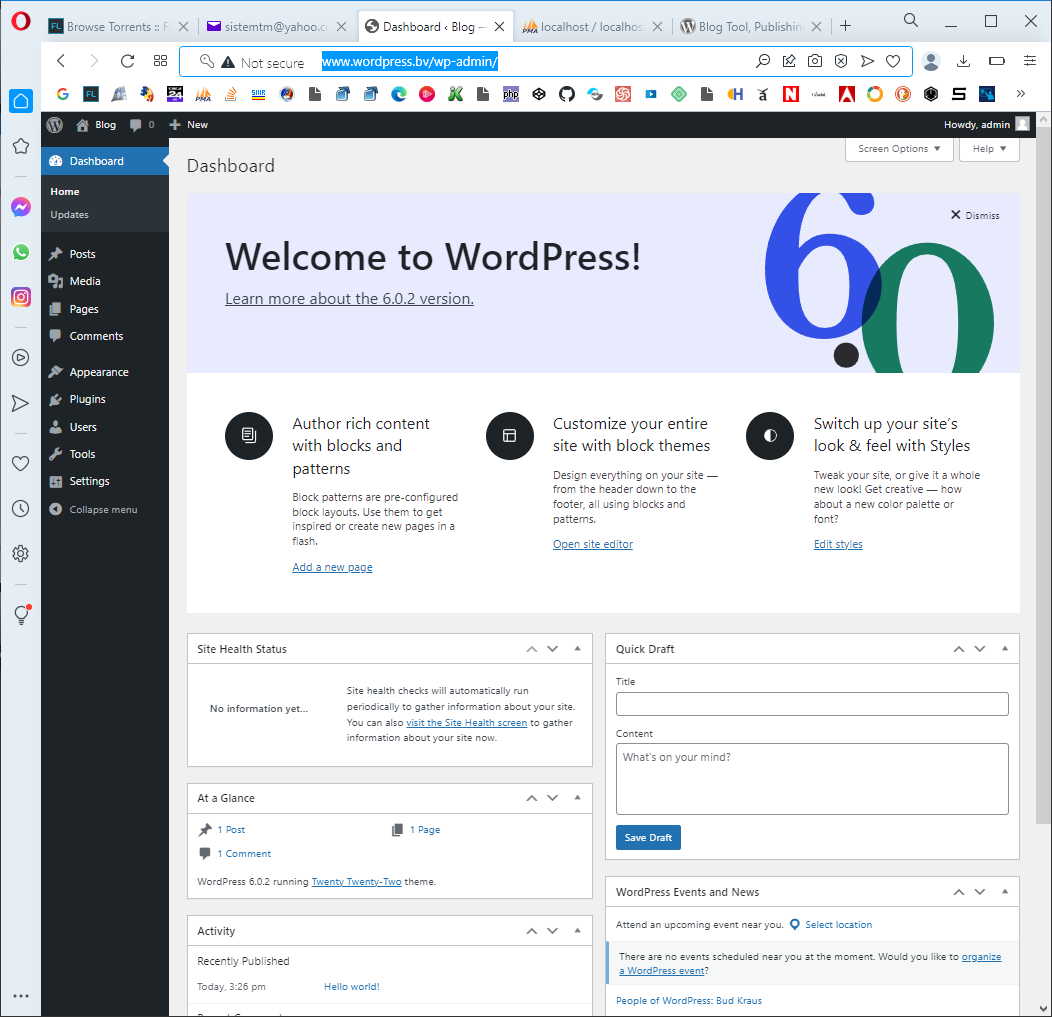
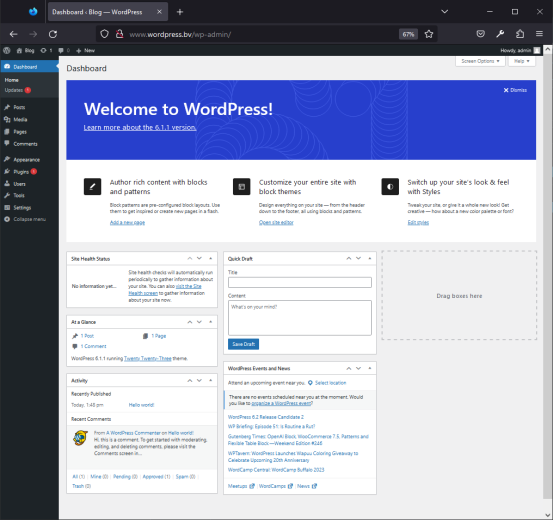
* + Success

Log in

* + W
    - Username or email address => admin
    - Password => wVw1C!f1C8glwxamE9

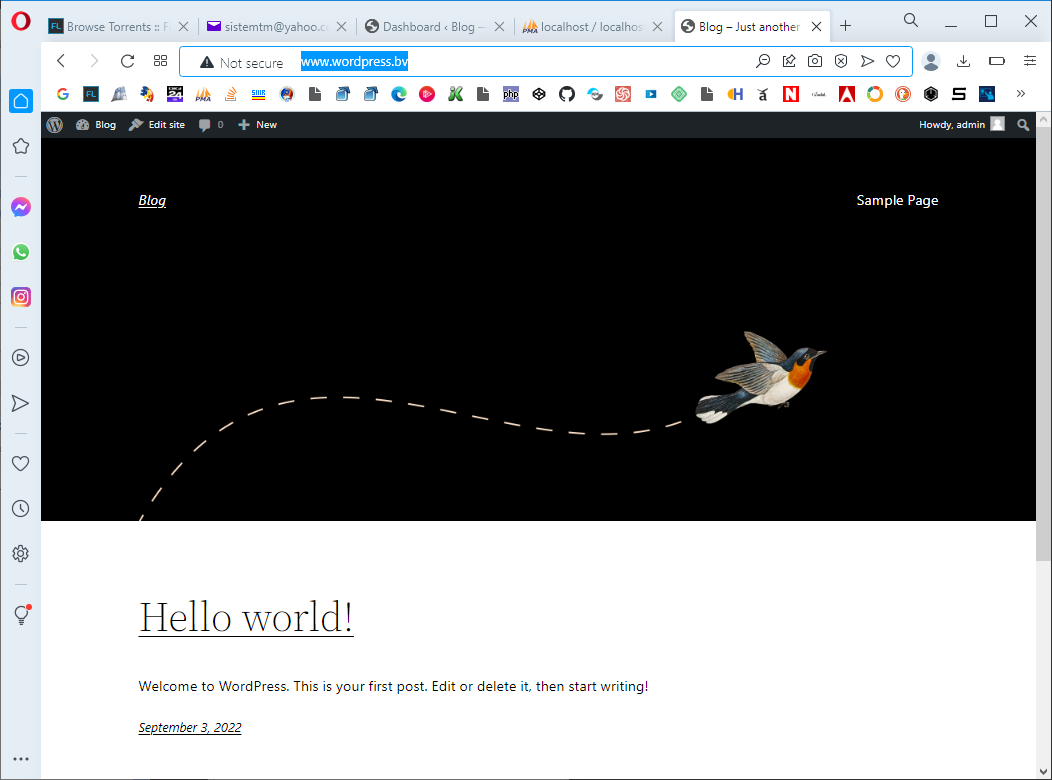
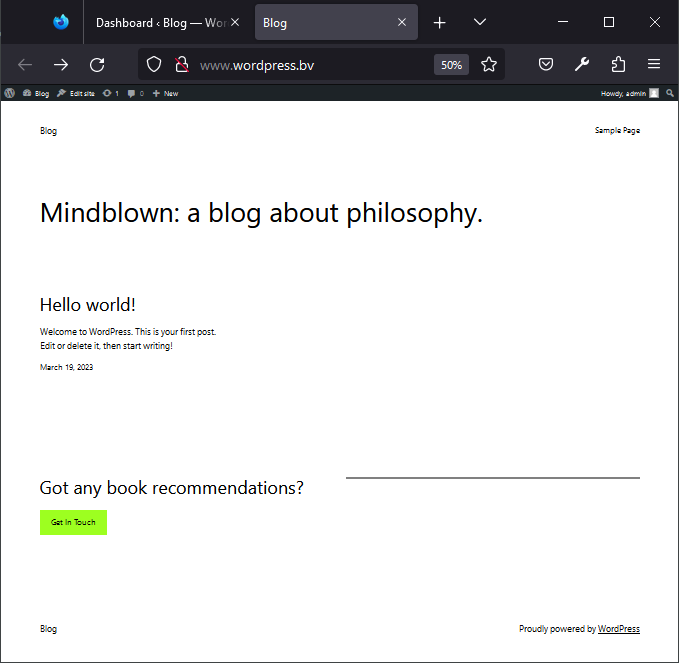
Log in

* <http://www.wordpress.bv/wp-admin/>

* Vizualizarea site-ului creat:

<http://www.wordpress.bv>

* Logare la Dashboard:.

[www.wordpress.bv/wp-login.php](http://www.wordpress.bv/wp-login.php)

* + - * Personalizare Dashboard:
* Screen Options:
  + Site Health Status
  + At a Glance
* Settings:
  + General Settings
    - Site Title => Food Blog
    - Tagline => For the love of Food
    - Membership => Anyone can register
    - Timezone => UTC+2
    - Time Format => 3:58 PM
  + Discussion Settings
    - Comments should be displayed with the  **newer** comments at the top of each page
    - Default Avatar => Gravatar Logo
  + Permalink Settings
    - Common Settings
      * ~~Plain~~ [~~http://www.wordpress.bv/?p=123~~](http://www.wordpress.bv/?p=123)
      * Post name <http://www.wordpress.bv/sample-post/>
* Plugins
  + Add New
    - Classic Editor → Install Now

# Brad Schiff „Become a WordPress Developer.. Unlocking Power With Code” [Udemy] (2022)

Site-uri oficiale WordPress:

https://codex.wordpress.org

<https://developer.wordpress.org>

Cod HTML static pentru pornirea unui site PHP dynamic:

<https://github.com/LearnWebCode/university-static>

* + - Instalarea unui mediu de dezvoltare local:

<https://localwp.com> 🡪 Download

=> local-6.7.0-windows.exe

=> Choose installation options:

Anyone who uses this computer (all users)

=> Destination Folder

C:\Program Files (x86)\Local

* + - Configurare inițială:

I have read and agree to Local’s Terms of Service

=> Create a new site

What’s your site name? => Fictional University

Choose your environment => Preferred

Setup WordPress:

WordPress Username: => me

WordPress Password => qwert.0123

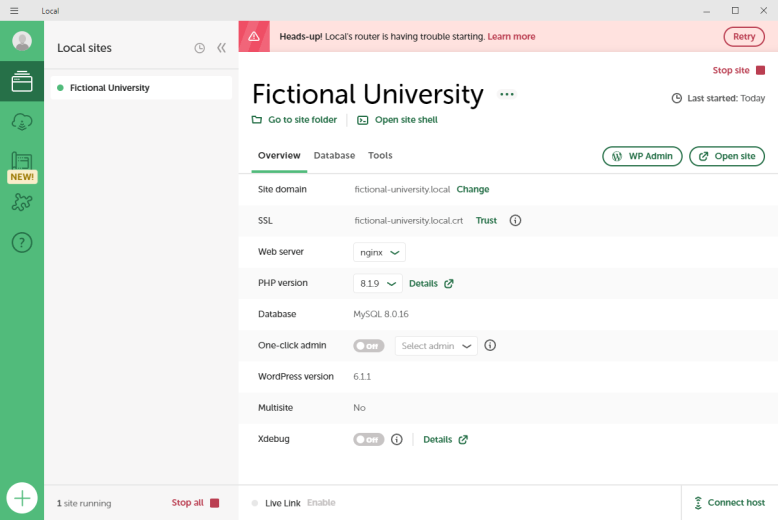
WordPress Email => [dev-email@wpengine.local](mailto:dev-email@wpengine.local)

Windows Defender Firewall has blocked some features of this app

=>  Public networks, such as those in airports and coffee shops

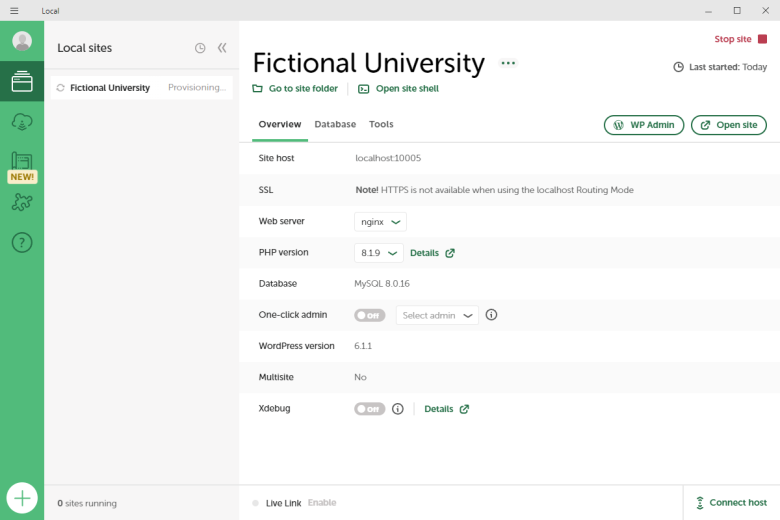
Allow access

=> Instalarea site-ului Fictional University cu butoanele ADMIN şi VIEW SITE

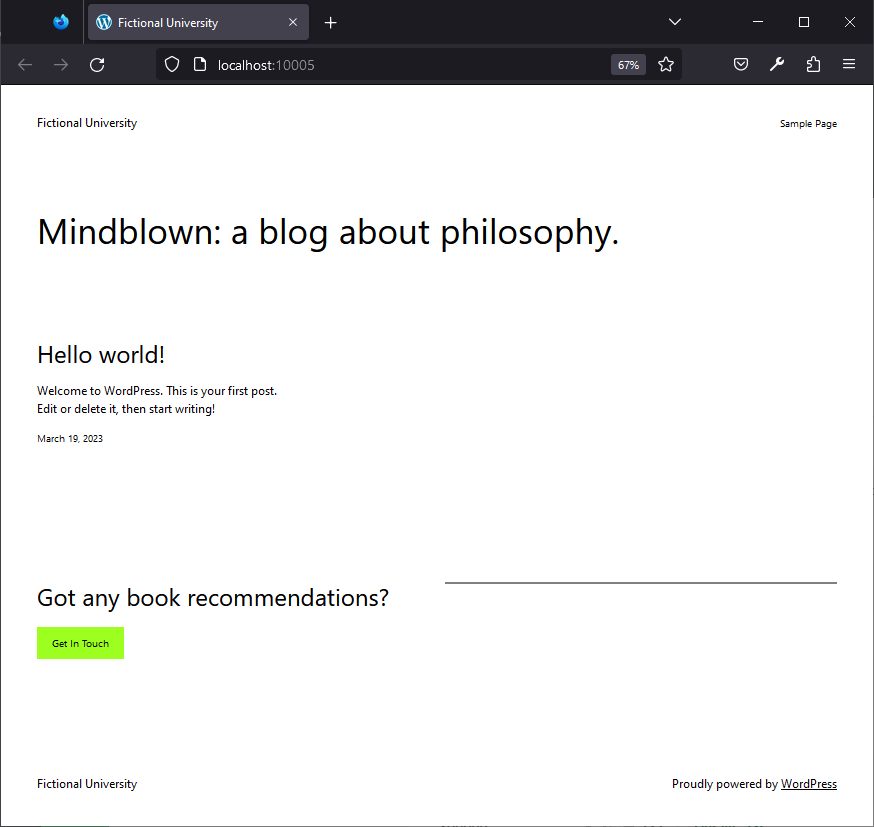


* + - Setarea ≡ > Preferences > Advanced > Router Mode >localhost

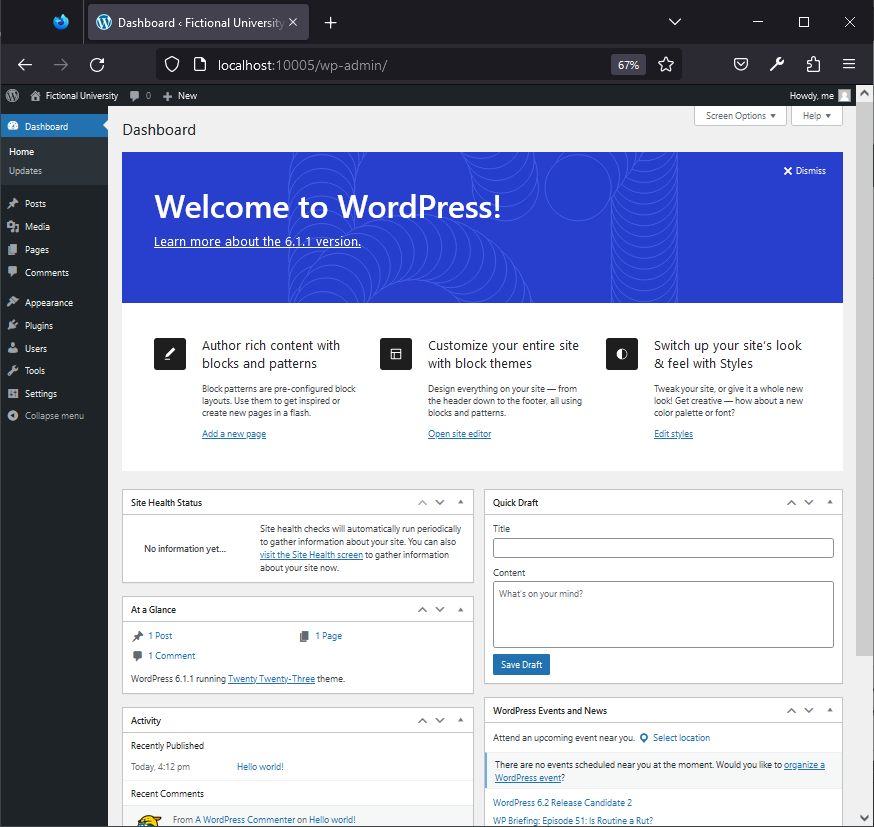
=> Dispare mesajul de eroare legat de router-ul local



* + - Site-urile nou create vor fi localizate în folder-ul C:\Users\me\Local Sites – locul se poate schimba din ≡ > Preferences > New site defaults > Sites path => E:\work\vhosts\www.local.bv\public\_html
    - Vizualizarea site-ului: butonul Open Site



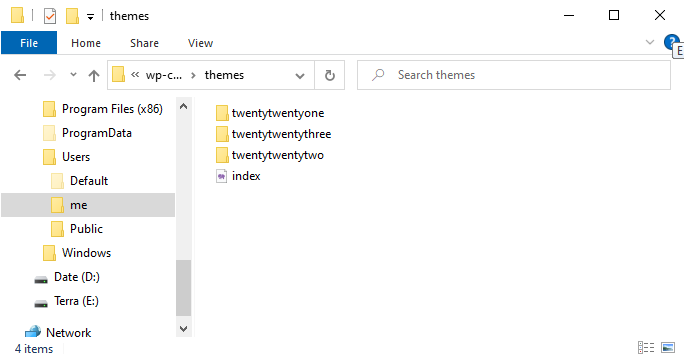
* Trecerea în browser în modul de administrare: <http://localhost:10005/wp-admin/>



* Fișierele sistem care susțin WordPress pentru site se află în folder-ul C:\Users\me\Local Sites\fictional-university\app\public.



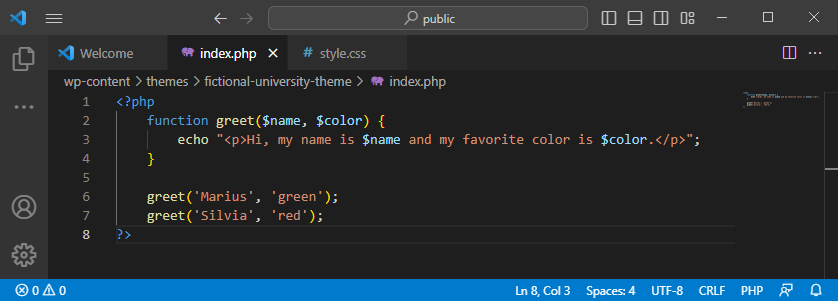
* În folder-ul C:\Users\me\Local Sites\fictional-university\app\public\wp-content\themes se află temele instalate



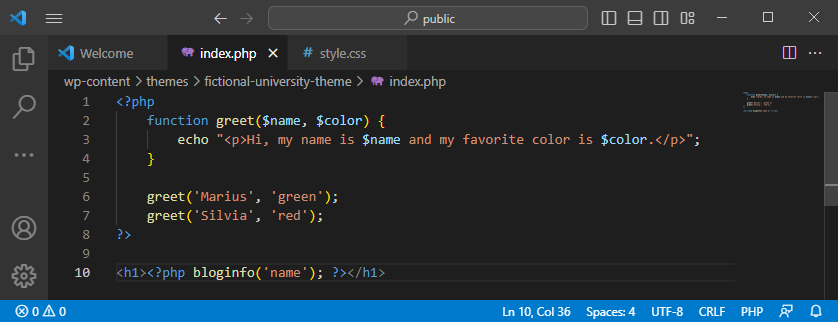
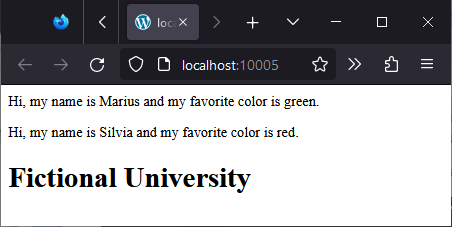
* Deoarece numele utilizatorului nu poate fi schimbat, pentru a afişa alt nume într-o pagină prin cod PHP, în pagina de administrare a site-ului, la secţiunea Users > me > Nickname se poate specifica numele respective, iar la Display name publicly as se poate alege ce nume să se afişeze.

## PHP

* Editorul de cod: Visual Studio Code, recomandat pentru că are o linie de comandă încorporată, care punctează automat la directorul poriectului curent.
* Se editeazăfișiere PHP în folder-ul C:\Users\me\Local Sites\fictional-university\app\public.
* Codul PHP se include între <?php şi ?>.
* Funciile se defines cu function() şi variabilele PHP cu semnul $ în faţă:



* Funcţii create şi definite în WordPress care pot fi preluate de PHP:
  + bloginfo(‘name’): afişează numele site-ului, care poate fi găsit în pagina de administrare, la secţiunea Settings > General > Site Title:

* + bloginfo(‘description): afişează numele site-ului, care poate fi găsit în pagina de administrare, la secţiunea Settings > General > Tagline.
* Tablourile (Arrays) se definesc cu array(). Sunt utile pentru ciclare, mai ales cu while, în WordPress. Lungimea unui tablou: count(nume\_tablou).

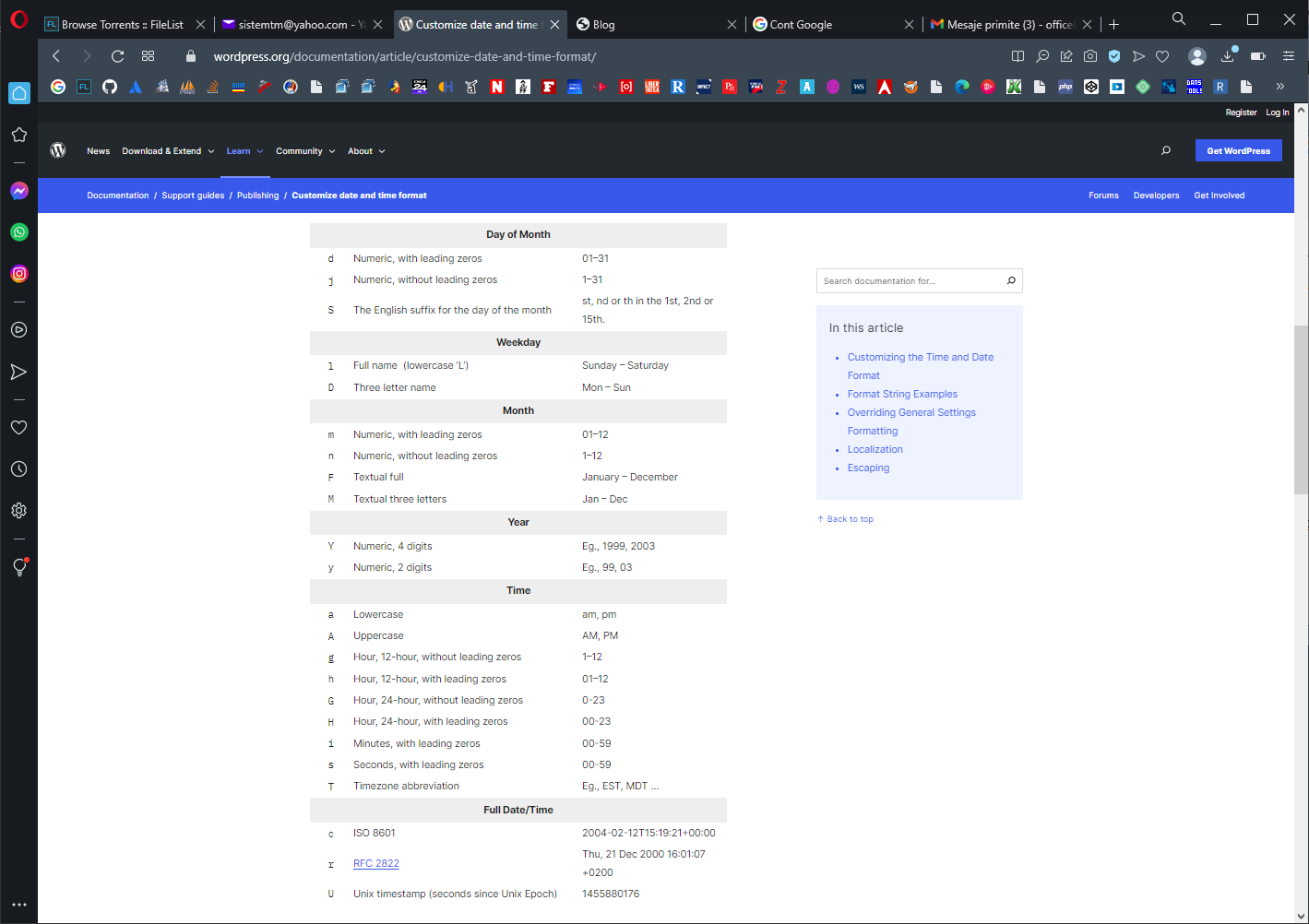
$nume\_variabilă = array(‘item1’, ‘item2’, …);

* Tablouri asociative:

$nume\_variabilă = array(‘item1’ => ‘val1’, ‘item2’ => ‘val2’, …);

* Convenţie funcţii: dacă numele unei funcţii începe cu get, atunci returnează un rezultat, iar cele care încep cu the afişează rezultatul, echivalentul lui echo.
* Funcţia WordPress the\_time() returnează date despre data calendaristică şi oră şi poate fi formatată cu argumente descrise în referinţa funţiei de la <https://developer.wordpress.org/reference/functions/the_time/>, <https://wordpress.org/documentation/article/customize-date-and-time-format/> :

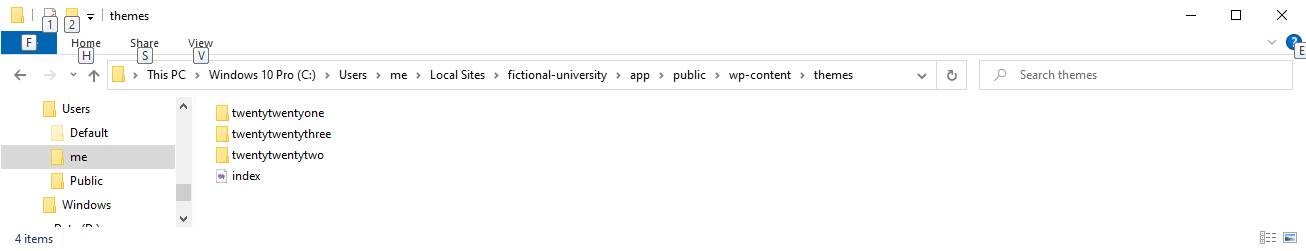
the\_time(‘d.m.Y’) => 25.03.2023.



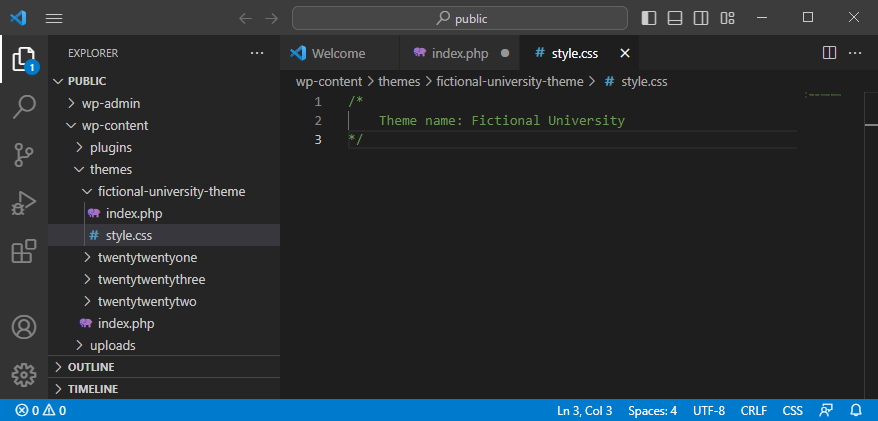
* Funcţia site\_title() returnează adresa site-ului, iar dacă i se furnizează ca argument ‘/valoare\_slug’ returnează link-ul paginii cu slug-ul respectiv.
* Funcţia wp\_trim\_words(textul\_de\_trunchiat, limita\_de\_cuvinte) trunchiază un text la un număr specificat de cuvinte.
* Funcţia get\_post\_type() poate returna valoarea post sau page.
* Funcţia has\_excerpt() returnează true dacă postarea are setat un excerpt propriu, scris manual în pagina de administrare, iar get\_the\_excerpt() returnează valoarea propriu-zisă.
* Funcţia the\_excerpt() determină trecerea pe rândul următor.

## Teme (Themes)

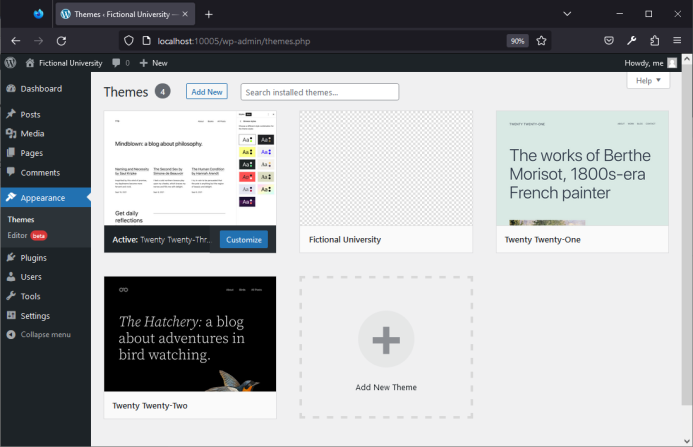
* Locație: C:\Users\me\Local Sites\fictional-university\app\public\wp-content\themes, adică în folder-ul „app\public\wp-content\themes” al site-ului.



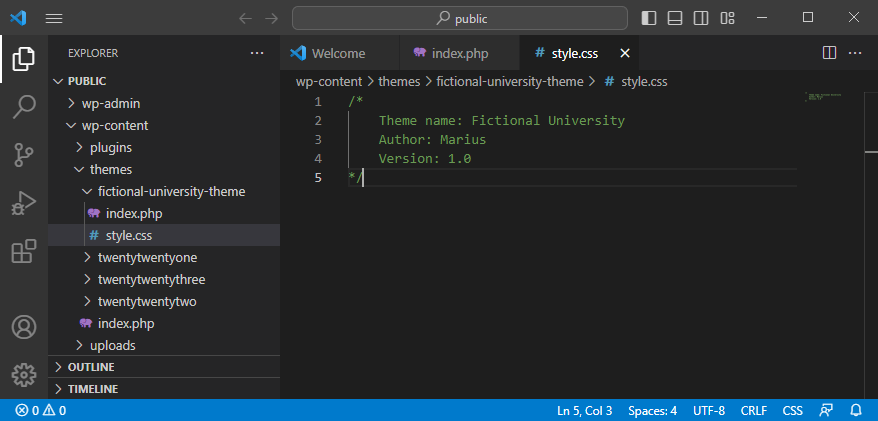
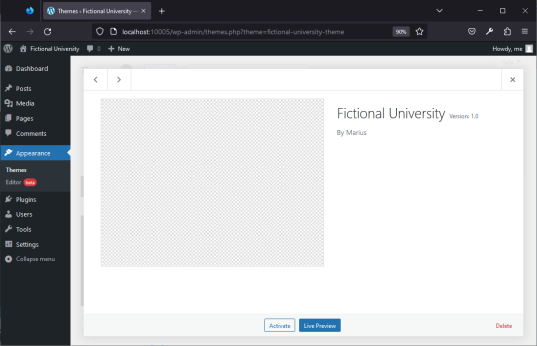
* Crearea unei teme noi: crearea unui folder nou, care să cuprindă ce puţin două fişiere, index.php şi style.css.
* Fişierul style.css mai are rolul de a furniza informaţii despre temă şi şi despre modul în care se furnizează WordPress – la începutul fişierului.



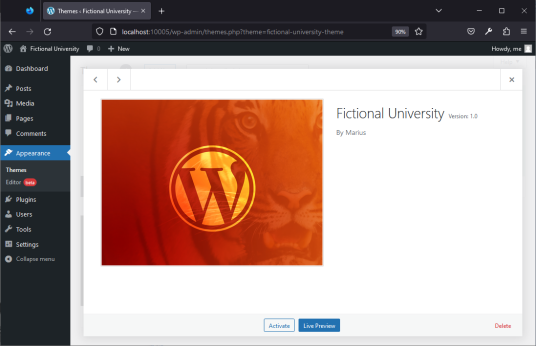
* Reîmprospătarea panoului de administrare afişează tema nou creată:

* În fişierul style.css, se pot completa date noi despre numele autorului şi versiune.

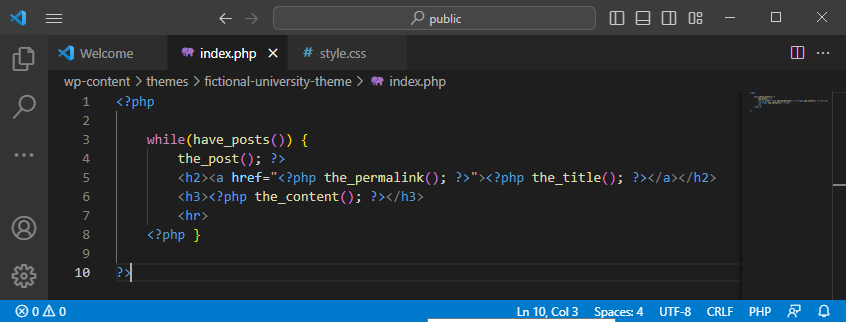
* Pentru o imagine de previzualizare a site-ului, se adăugă fişierul respectiv în folder-ul themes şi se redenumeşte „screenshot.png”.



* Dimensiunea ideală a imaginii: 1200 x 900 pixeli.
* Activarea temei: butonul Activate. Va fi afişat site-ul folder-ului temei respective.

## Postări (Posts)

* Sunt în pagina de administrare, secţiunea Posts.
* Creare postare: butonul Add New.
* Vizualizarea postărilor:



* Funcţia WordPress the\_post() iniţializează tratarea postărilor din site-ul WordPress activ.
* Pentru utilizarea adresei unei postări, se folosește funcţia WordPress the\_permalink(). Se mai numeşte şi slug.
* Dacă se crează un fişier single.php în partea principală a site-ului, atunci slug-ul va duce la acea pagină. Acest fişier este folosit numai pentru postări individuale.
* Construcţia „while(have\_posts()) {}” se numeşte Ciclul (The Loop) în WordPress şi este inima şi sufletul acestuia.
* Crearea unei categorii pentru postări se face în pagina de administrare WordPress prin intrarea pe o postare şi, în secţiunea Post > Categories se alege butonul Add New Category.
* În secţiunea Post > Excerpt > WRITE AN EXCERPT (OPTIONAL) se poate scrie textul personalizat pentru un excerpt.
* Pentru a fi afişat câmpul excerpt, trebuie selectată opţiunea Excerpt în secţiunea Screen Options a paginii de administrare.

## Antete şi subsoluri (Headers and Footers)

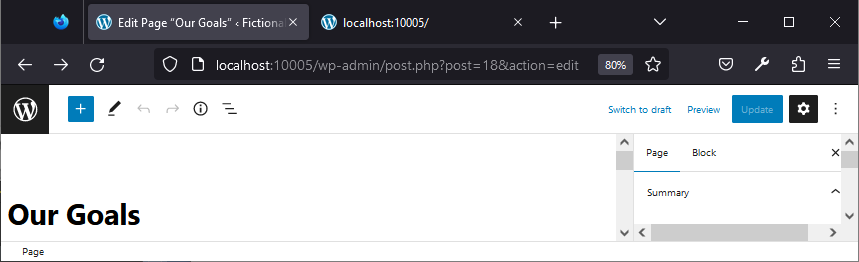
* Se creează fişierele header.php şi footer.php în partea principală a site-ului pentru antete şi subsoluri globale.
* Se folosesc funcţiile get\_header() şi get\_footer().
* Funcţia WordPress get\_head() încarcă fişierul header.php.
* Funcţia WordPress wp\_footer() încarcă bara de navigare superioară în pagina principală a site-ului, corespunzătoare lui index.php. Se include în fişierul footer.php, pe ultima linie din corpul body.

## Fişierul functions.php

* Conţine funcţia WordPress add\_action(), cu două argumente, primul spune ce tip de instrucţiuni se încearcă a fi executate, şi al doilea este numele unei funcţii ce se doreşte a fi executată.
* WordPress permite furnizarea de instrucţiuni prin această funcţie, pentru a-i spune ce să facă.
* Argumentul wp\_enqueue\_scripts pe prima poziţie este pentru a încărca un fişier, CSS sau JavaScript.
* Funcţia WordPress wp\_enqueue\_style() necesită două argumente, primul, pentru un alias pentru fişierul CSS principal, şi al doilea, pentru locaţia fişierului respectiv.
* Funcţia WordPress get\_stylesheet\_uri() furnizează fişierul CSS din locaţia site-ului.
* Funcţia WordPress wp\_enqueue\_script() se foloseşte pentru fişiere JavaScript. Are cinci argumente, al treilea fiind un tablou array, care specifică dacă script-ul depinde de alte script-uri, de obicei jquery, sau NULL, al patrulea fiind numărul variantei, iar ultimul fiind true dacă se doreşte încărcarea script-ului înainte de tag-ul de închidere din body.
* Funcţia WordPress add\_theme\_support(‘title-tag’) determină numele paginii afişat în browser prin combinarea titlului paginii din panoul de administrare cu – şi cu titlul site-ului.

## Pagini (Pages)

* Sunt în pagina de administrare, secţiunea Pages.
* Creare pagină: butonul Add New.
* Pentru afişarea de pagini individuale cu construcţia have\_posts(), WordPress foloseşte fişierul page.php.
* La crearea unei pagini, WordPress crează automat o pagină de confidenţialitate, Privacy Policy, care se publică cu butonul Publish.
* Funcţia WordPress site\_url(‘title-tag’) returnează link-ul pentru pagina cu slug-ul repectiv. fără argument navighează la pagina iniţială, Home page.
* La crearea unei pagini, în secţiunea Page > Page Attributes > PARENT PAGE, se poate specifica pagina părinte.
* În bara de adrese a browser-ului, după denumirea fişierului .php, poate să apară semnul întrebării, urmat de cuvântul „post”, apoi de „=”, şi de un număr, număr care este identificatorul numeric al paginii



* Funcţia WordPress ce returnează acest identificator numeric unic al paginii curente care este vizualizată este get\_the\_ID().
* Funcţia WordPress ce returnează identificatorul numeric al paginii părinte este wp\_get\_post\_parent\_id(număr\_pagină\_curentă).
* Pentru lucru dinamic, se foloseşte construcţia wp\_get\_post\_parent\_id(get\_the\_ID()).
* Dacă pagina curentă nu are o pagină părinte, numărul este 0, altfel este un număr întreg mai mare ca zero.
* Funcţia WordPress get\_the\_title(număr\_pagină) returnează titlul paginii cu numărul respectiv, pe când funcţia the\_title() pe al pagini curente.
* Funcţia WordPress get\_permalink(număr\_de\_identificare) va returna permalink-ul pentru pagina sau postarea respectivă.
* Funcţia WordPress wp\_list\_pages() crea link-uri către toate paginile site-ului, fără nici un argument, dar argumentul care i se poate transmite trebuie să fie un tablou asociativ.
* Pentru a vedea dacă o pagină este părinte, se poate verifica dacă acea pagină are copii prin

<?php get\_pages(array(

‘child\_of’ => get\_the\_ID()

?>

* Sortarea afişării paginilor copii se face, implicit, în ordine alfabetică. Ordinea se poate stabili prin secţiunea Page > Page Attributes > ORDER.

## Recomandări

* Pentru adaptarea site-ului web la dimensiunea dispozitivului, se adaugă un tag în secţiunea head:

<meta name=”viewport” content=”width=device-width, initial-scale=1”>

* Pe smartphone-uri se obţine o vizualizare pe o singură coloană şi bara de navigare este redusă la trei linii orizontale suprapuse, dar nu acest tag face site-ul web responsive – este nevoie de cod pentru acest lucru.
* Acest tag spune doar dispozitivelor să folosească mărimea lor naturală.
* Pentru a comunica browser-elor web ce limbă foloseşte site-ul, se poate stabili în WordPress, în secţiunea Settings > SiteLanguage, şi se poate folosi cod PHP:

<html <?php language\_attributes(); ?>

* Acest lucru va genera cod HTML <html lang=”en-us”>.
* Pentru setul de caractere se poate folosi PHP într-un tag meta:

<meta charset=”<?php bloginfo(‘charset’); ?>”>

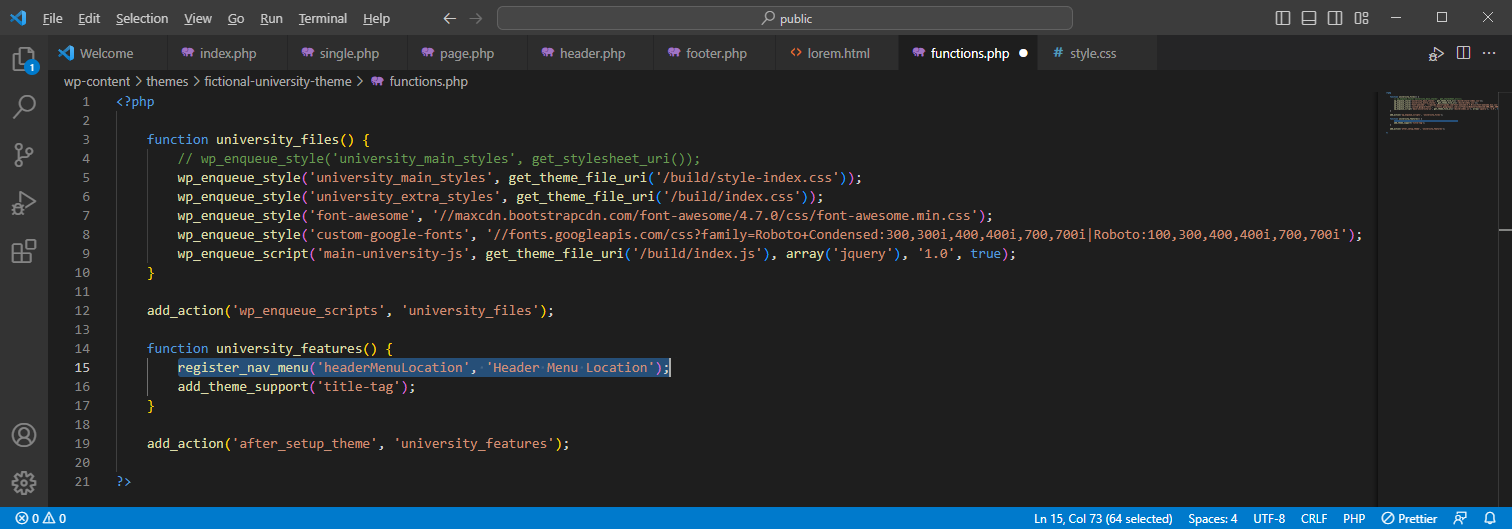
* Se generează codul HTML <meta charset=”UTF-8”>.
* Prin folosirea codului PHP în tag-ul body

<body <?php body\_class(); >>

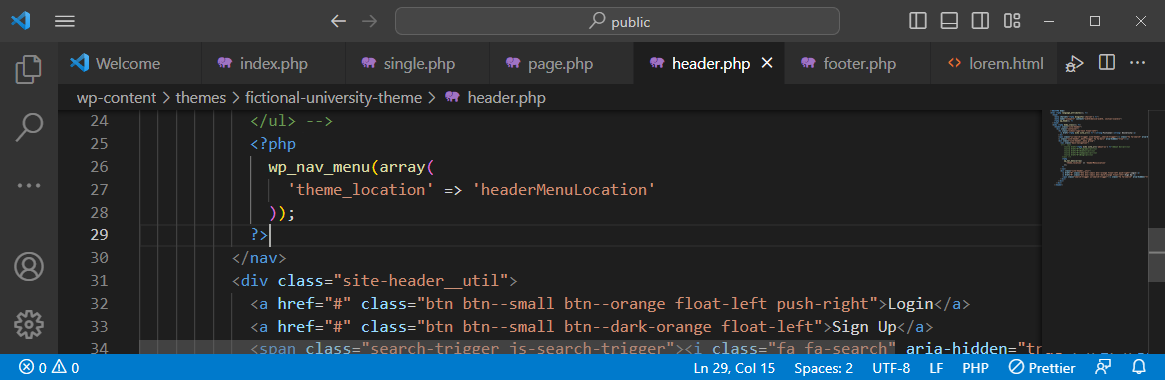
în codul HTML sunt generate mai multe clase în funcţie de contextul paginii.

## Meniuri de navigare dinamice

* Se pot face din pagina de administrare WordPress, secţiunea Appearance > Menus, dar înainte trebuie înregistrate locaţiile meniului în cadrul temei curente care se construieşte.
* Înregistrarea se face prin funcţia WordPress register\_nav\_menu(), cu două argumente, primul fiind numele de cod al meniului,.funcţie înscrisă în interiorul altei funcţii, apelată de funcţia add\_action() în interiorul fişierului functions.php.



* Al doilea argument este cel care apare afişat în pagina de administrare, la Appearance > Menus > Menu Settings > Display location >  al doilea argument de la funcţia register\_nav\_menu().
* Un meniu este afişat prin funcţia wp\_nav\_menu(), cu argumentul un tablou asociativ, punctând la numele de cod al meniului.



* Funcţia WordPress is\_page(‘valoare\_slug’) returnează true dacă pagina cu slug-ul respectiv este curentă.
* Funcţia WordPress wp\_get\_post\_parent\_id(get\_the\_ID()) returnează identificatorul numeric post al paginii părinte pentru pagina curentă, dată de argumentul get\_the\_ID(), argument care poate fi înlocuit cu 0, şi WordPress va şti că se face referire la pagina curentă.

## Aspectul de blog al site-ului web

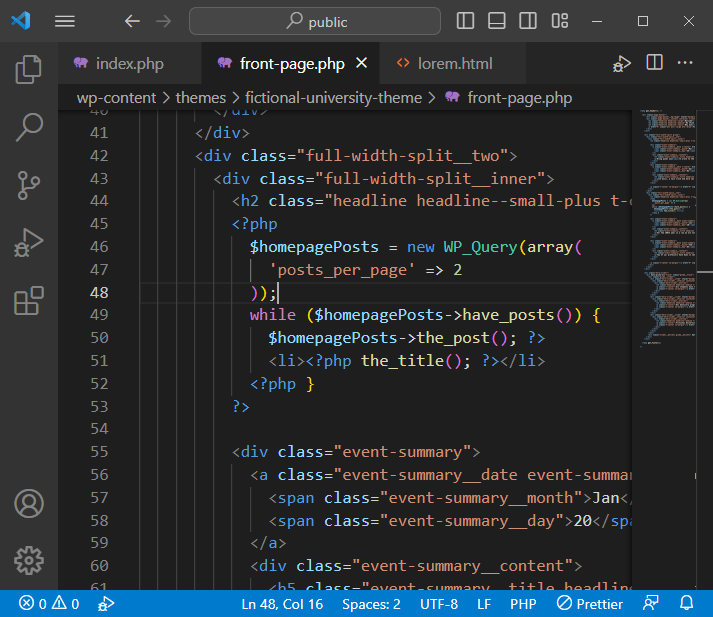
* Implicit, WordPress face pagina de home o listare generică de blog – afişează toate postările de blog mai recente.
* Dacă s-a creat o temă nouă, fără pagină de blog, se poate crea una din pagina de administrare WordPress.
* În secţiunea Settings > Reading > Front page displays se poate specifica ce să afişeze pagina din faţă, de home – ultimele postări sau o pagină statică, una pentru home şi una pentru postări.
* În rădăcina site-ului web se creează un fişier front-page.php, care va conţine codul pentru pagina de home, iar fişierul iniţial index.php va fi responsabil de ecranul paginii de listare de blog generic.
* WordPress afişează implicit cele mai recente zece postări de blog – secţiunea Settings > Reading > Blog pages show at most 10 post.
* Funcţia WordPress paginate\_links() returnează legături la pagini pentru postări care depăşesc numărul maxim setat în pagina de administrare – se numesc legături de paginare.
* Fişierul single.php conţine şablonul pentru pagina unei singure postări – este pentru postări individuale, la fel cum fişierul page.php este pentru pagini individuale.

## Ecrane de arhive

* Sunt pagini afişate când se selectează o legătură dintr-o pagină de blog către autor, categorie,dată calendaristică, şa, care afişează numai postările ce corespund legăturii respective – la sfârşitul barei de adrese vor apărea /category/nume\_categorie sau /author/nume\_autor, paginile respective bazându-se iniţial pe fişierul index.php, dacă nu există fişierul responsabil archive.php.
* Funcţiile single\_cat\_title() şi the\_author() afişează numele categoriei sau al autorului.
* Postările de blog arhivate după date calendaristice sunt afişate prin specificarea unui tip de dată în bara de adrese după adresa site-ului (../2023/03 va determina afişarea postărilor din luna martie 2023) – pot fi arhive după ziuă, lună sau an
* Funcţia nouă WordPress the\_archive\_title() simplifică lucrurile şi afişează titlul arhivei sub forma denumire\_arhivă: valoare\_arhivă.
* Funcţia WordPress soră the\_archive\_description() afişează informaţia setată în pagina de administrare Wordpress:
  + pentru autor: secţiunea Users > Profile > Biographical info
  + pentru categorie: secţiunea Posts > Categories > nume\_categorie > Description.

## Interogări personalizate

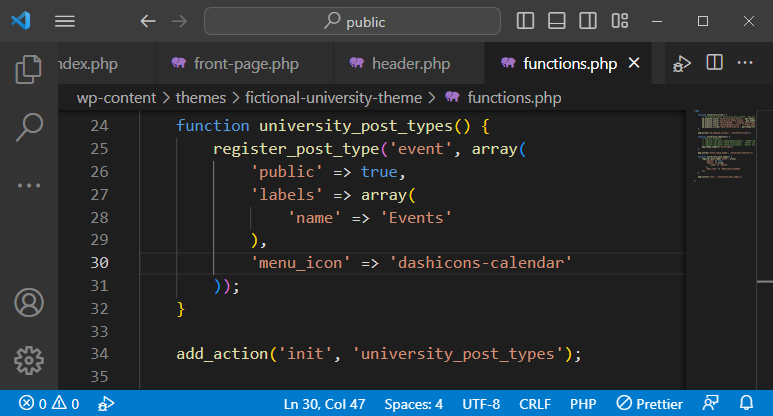
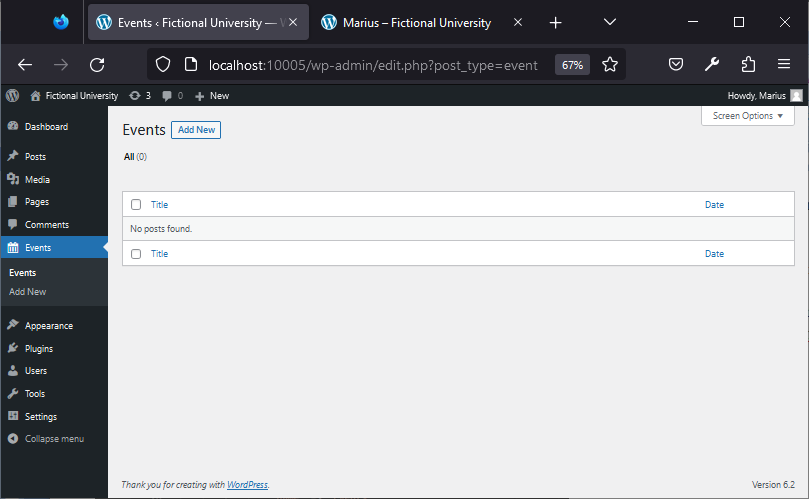
* Interogările normale (implicite) ale bazei de date se fac automat de către WordPress pe baza specificului URL-ului vizitat.
* Interogările personalizate permit încărcarea a orice, oriunde.
* Înaintea creării unei interogări personalizate, se creează o variabilă care să reprezinte numele interogării cu ajutorul construcţiei new WP\_Query() – se creează un obiect din clasa WP\_Query.
* Argumentul este un tablou asociativ, care personalizează interogarea cu diferiţi parametri.
* Apoi, se utilizează numele interogării în faţa funcţiilor implicite.



* După încheierea ciclului unei interogări personalizate, se recomandă apelarea funcţiei wp\_reset\_postdata() pentru a reseta diferite date WordPress şi variabile globale înapoi la starea care erau înainte.

## Tipuri de postări personalizate

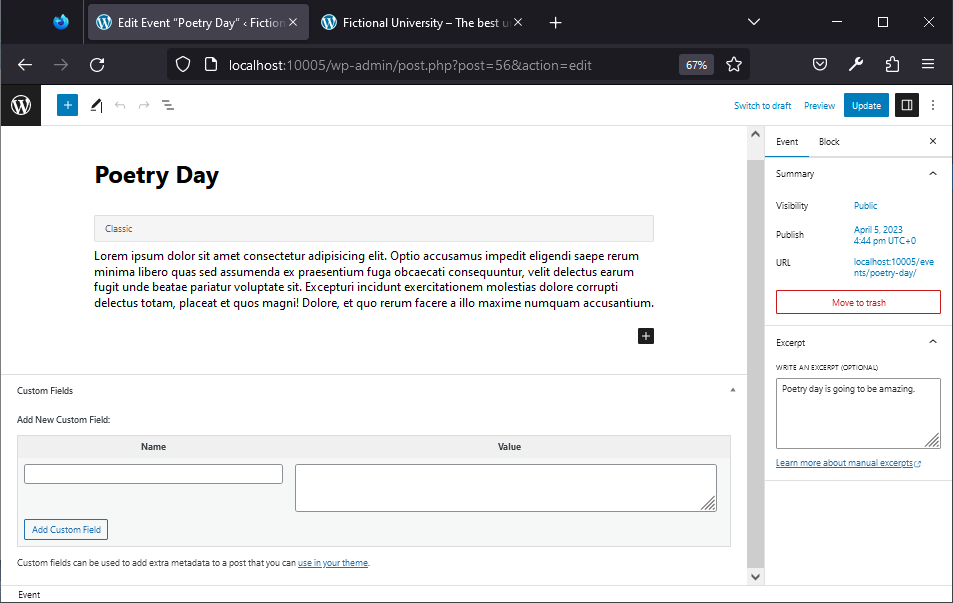
* WordPress vine implicit numai cu două tipuri de postări, postări şi pagini.
* O pagină este doar o postare cu un tip de postare special al unei pagini – în panoul de administrare se pot selecta pagini, iar URL-ul lor se termină cu „wp\_admin/edit.php?post\_type=page”.
* Toţi itemii sau intrările de conţinut în WordPress sunt, în realitate, doar postări cu diferite etichete, pentru diferite tipuri.
* Pentru crearea unui tip nou de postare se foloseşte funcţia add\_action(), cu argumentele ‘init’, pentru apelarea la momentul iniţial, şi numele funcţiei propriu zise care va crea postarea, ex. add\_action(‘init’, ‘university\_post\_types’).
* Funcţia care va crea postarea apelează funcţia WordPress register\_post\_type(), cu două argumente, tipul postării şi un tablou asociativ care conţine parametrii ce definesc postarea.
* Parametrul ‘public’ => true face tipul de postare vizibil pentru editare şi vizualizare.
* Astfel, în partea laterală din stânga a paginii de administrare apare un rând nou, numit tot Post şi cu acelaşi tip de iconiţă, dar de tipul cel nou.
* Pentru schimbarea denumirii afişate se foloseşte parametrul ‘labels’ => array (‘name’ => ‘denumire nouă’).
* Pentru schimbarea iconiţei, se foloseşte parametrul ‘menu\_icon’, cu o referire la o denumirea unei iconiţe din site-ul developer.wordpress.org/resource/dashicons/.

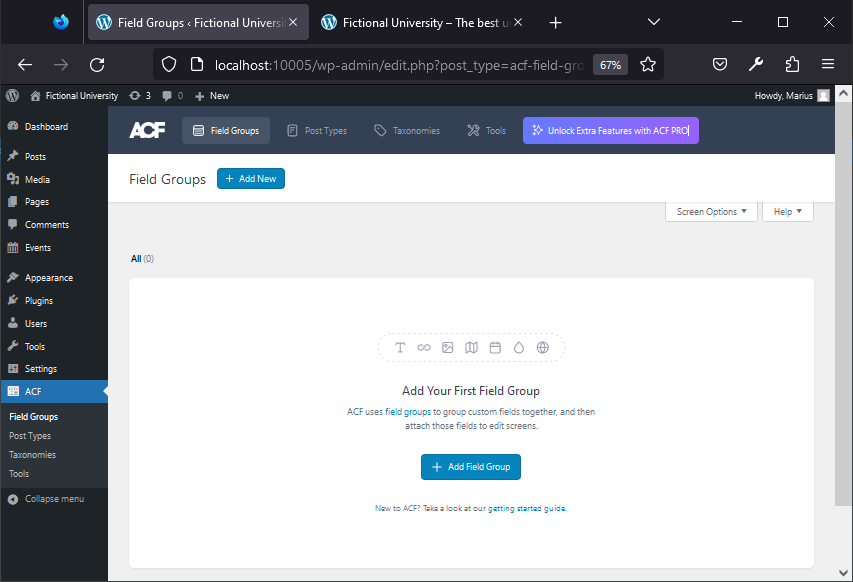
* Pentru a avea un cod care să fie accesibil şi executat indiferent de tema utilizată, se creează un folder mu-plugins alături de folder-ul themes – toate fişierele .php de aici de execută automat pentru toate temele utilizate, spre deosebire de un plugin care poate fi activat/dezactivat.
* Crearea unui articol într-o postare personalizată se face în pagina de administrare prin meniul din stânga Add New sau prin butonul de sus tot Add New.
* După crearea unei postări personalizate, WordPress nu face actualizarea automată a structurii de permalink-uri, ci trebuie făcută manual din pagina de administrarea prin secţiunea Settings > Permalinks > Save changes, chiar dacă nu s-a făcut nici o schimbare – se obţine mesajul **Permalink structure updated.**
* În plus, trebuie actualizat şi şablonul paginii pentru această postarizare personalizată, deoarece este folosit cel din fişierul single.php, responsabil pentru postările de blog, deci se va crea şablonul respectiv printr-un fişier .php, cu un nume special, alcătuit din single- şi numele tipului de postare personalizată – de ex. single-event.php pentru o postare de tip event.
* La înregistrarea acestui tip de postare personalizată, mai trebuie specificat pentru WordPress şi că noul tip de postare suportă o arhivare – prin adăugarea unui nou argument funcţiei register\_post\_type, şi anume ‘has\_archive’ => true, urmată de actualizarea structurii de permalink-uri.
* Tot prin funcţia register\_post\_type se poate schimba şi numele slug-ului pentru arhiva de articole ale acelui tip de postare personalizată, prin argumentul rewrite – de ex. pentru tipul event se poate folosi ‘rewrite’ => array(‘slug’ => ‘events’).
* Formatarea implicită a unei arhive este descrisă de fişierul archive.php
* Schimbarea formatării paginii arhivei pentru o postare personalizată se face prin crearea unui fişier .php cu numele archive-, urmat de tipul acelei postări – de ex. archive-event.php.
* Funcţia get\_post\_type\_archive\_link(‘numele\_tipului\_postării’) returnează slug-ul arhivei tipului de postare.
* Pentru specificarea faptului că o postare personalizată suport un câmp de excerpt, se adaugă un argument nou funcţiei register\_post\_type(): ‘supports’ => array(‘title’, ‘editor’, ‘excerpt’).
* Pentru utilizarea editorului modern în cazul unei postări personalizate, se adaugă argumentul ‘show\_in\_rest’ => true funcţiei register\_post\_type(), deoarece acest editor se bazează exclusiv pe JavaScript şi trebuie setat ca el să se afişeze în API-ul Rest. În plus, trebuie să fie specificat şi editor la argumentul support.

## Câmpuri personalizate

* În primul rând, argumentul supports al funcţiei register\_post\_type() primeşte un nou parametru ‘custom-fields’, de ex. ‘supports’ => array(‘title’, ‘editor’, ‘excerpt’, ‘custom-fields’).
* Apoi, se deschide o postare personalizată în pagina de administrare, se selectează meniul celor trei puncte din colţul dreapta-sus, ⁝ şi se alege opţiunea Preferences > Panels > Custom fields şi se activează butonul Enable & Reload.
* Va rezulta opţiunea pentru câmpuri personalizate în fereastra postării respective:



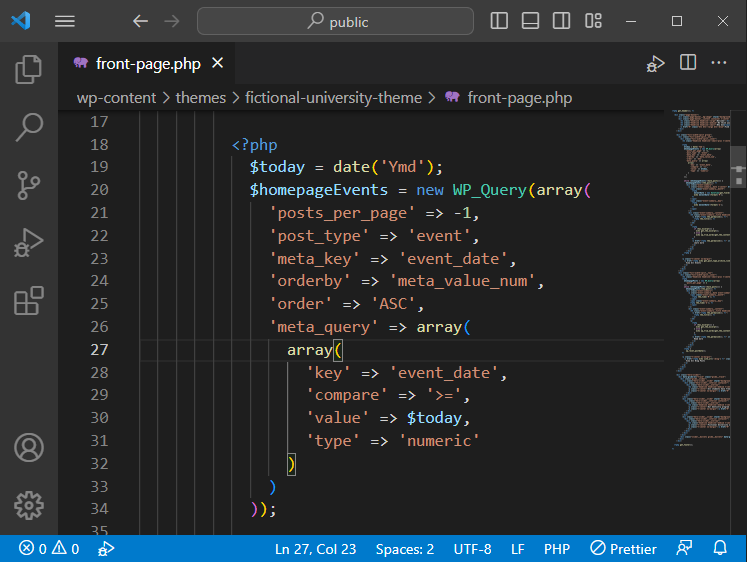
* Se pot folosi şi plugin-uri standard pentru câmpuri personalizate, Advanced Custom Fields (ACF) sau Custom Metaboxes2 (CMB2) – se va şterge opţiunea custom-fields din funcţia register\_post\_type().
* După instalarea şi activarea plugin-ului ACF, în banda laterală din stânga apare opţiunea ACF.



* Funcţia the\_plugin(‘nume\_câmp’) afişează valoarea câmpului, iar get\_field(‘nume\_câmp’) returnează această valoare.

## Ordonarea interogărilor personalizate – Sortarea

* Ordonarea implicită a postărilor este după data la care au fost create sau publicate, post\_date.
* La interogarea personalizată prin instanţierea clasei WP\_Query se poate adăuga parametrul ‘orderby’ => ‘numele\_câmpului’, ceea ce va determina sortarea în ordinea descrescătoare (DESC), care este cea implicită.
* Dacă se specifică ‘posts\_per\_page’ => -1, atunci se face referire la fiecare postare care îndeplineşte întreaga interogare deodată.
* Sortarea crescătoare se face prin parametrul ‘order’ => ‘ASC’.
* Parametrul ‘orderby’ => ‘rand’ determină sortarea aleatoare la fiecare reîmprospătare a browser-ului.
* În WordPress, datele meta sunt toate datele extra sau adiţionale personalizate asociate cu o postare.
* Pentru sortarea după un câmp personalizat, acesta se specifică în interogare prin parametrul meta\_key, iar sortarea se va face prin ‘orderby’ => ‘meta\_value’ pentru date de tip text şi prin ‘orderby’ => ‘meta\_value\_num’ pentru date numerice.
* O metainterogare se face prin parametrul meta\_query, cu un tablou asociativ, care poate conţine mai multe tablouri asociative, pentru mai multe criterii de interogare.

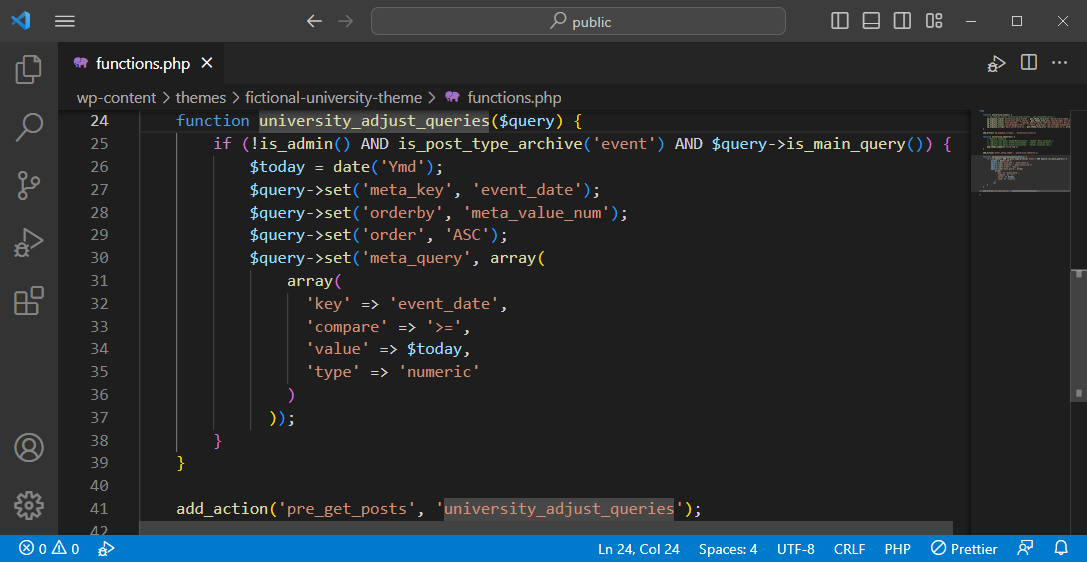


## Editarea interogărilor implicite

* Se adaugă o acţiune nouă în fişierul functions.php, referitoare la evenimentul WordPress ce are loc înainte de încărcarea postărilor, pre\_get\_posts, acţiune ce va apela o funcţie care va rafina interogarea implicită cu noi criterii.
* Atunci când WordPress va apela această funcţie, el îi va furniza interogarea implicită, care poate fi preluată ca parametru, astfel încât obiectul interogării să poată fi manipulat ulterior prin construcţia

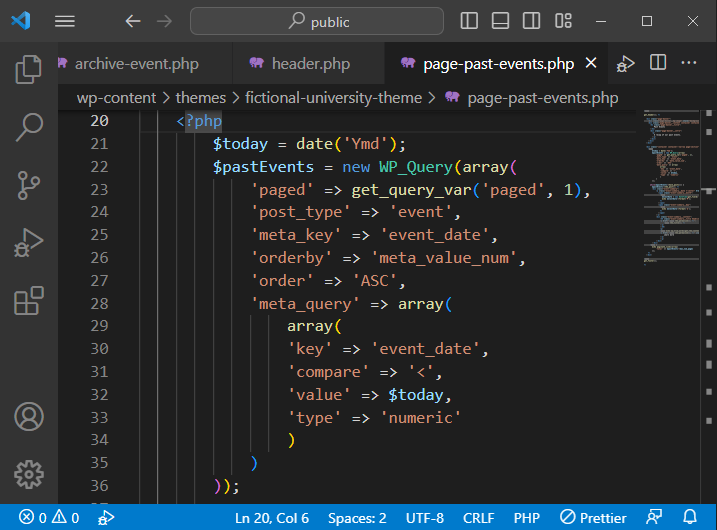
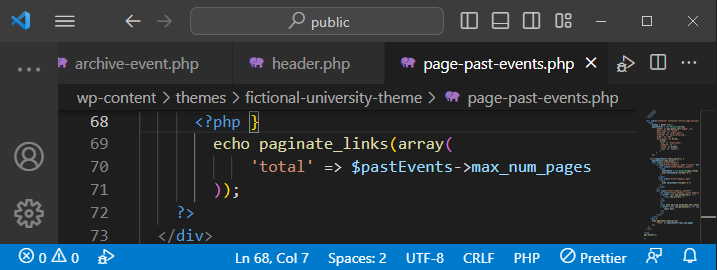
obiect->set(‘parametru\_interogare’, ‘valoare\_interogare’ ).

* Interogarea este atât de puternică, încât afectează şi postările din pagina de administrare, şi din toate celelalte pagini ale site-ului.
* Funcţia is\_admin() returnează true dacă pagina curentă este pagina de administrare, deci trebuie negată această pagină în condiţia de acces.
* Funcţia $interogare->is\_main\_query() returnează true dacă obiectul aparţine interogării personalizate, evitându-se, astfel, manipularea accidentală a altei interogări.



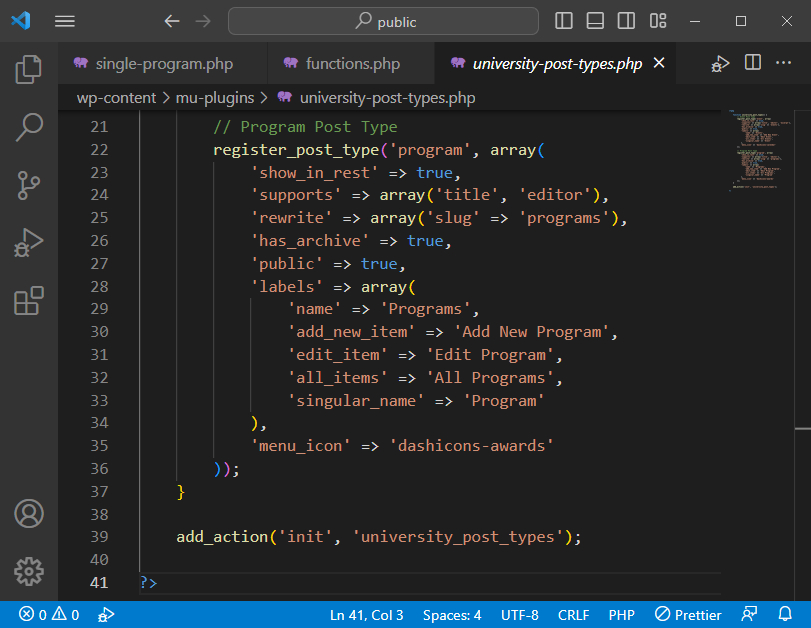
## Crearea unui şablon dedicat pentru o singură pagină

* Dacă se doreşte crearea unui şablon pentru o singură pagină, se creează în rădăcina site-ului un fişier .php cu numele page-valoarea\_slug\_pagină.
* În general, paginarea WordPress funcţionează numai cu interogările implicite pe care le face WordPress şi care sunt legate la URL-ul curent.
* În cazul interogărilor personalizate, trebuie să i se furnizeze funcţiei paginate\_links() un argument tablou asociativ cu atributul ‘total’ setat la numărul maxim de pagini al interogării respective.
* În plus, interogarea respectivă trebuie să conţină şi atributul ‘paged’ => get\_query\_var(‘paged’, 1).

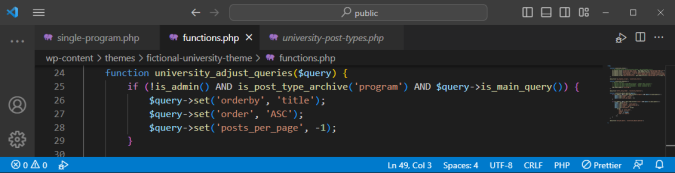


## Crearea de relaţii între conţinuturi ale diferitelor tipuri de postări

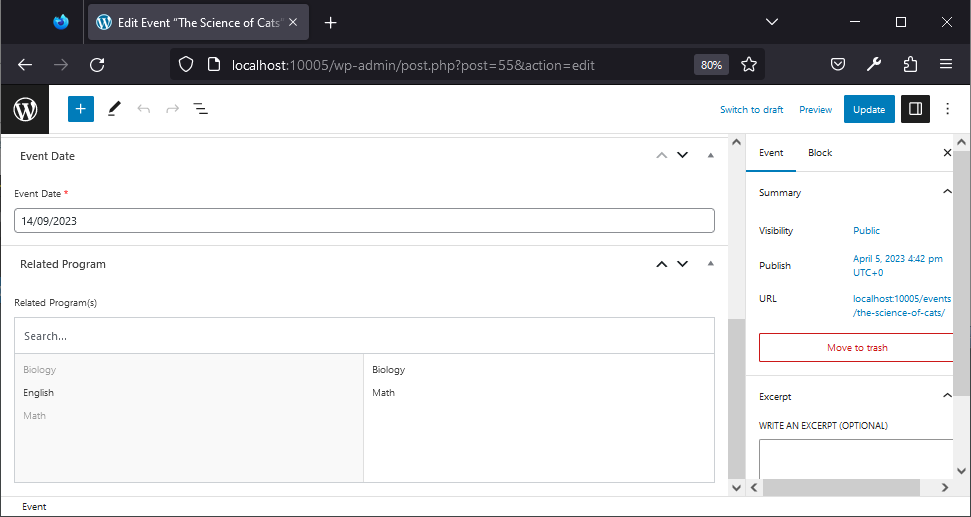
* Se creează un nou tip de postare, Program, pe lângă cel de Event, prin funcţia register\_post\_type(), iniţializată prin funcţia add\_action() din fişierul university-post-types.php din folder-ul mu-plugin din folder-ul wp-content.

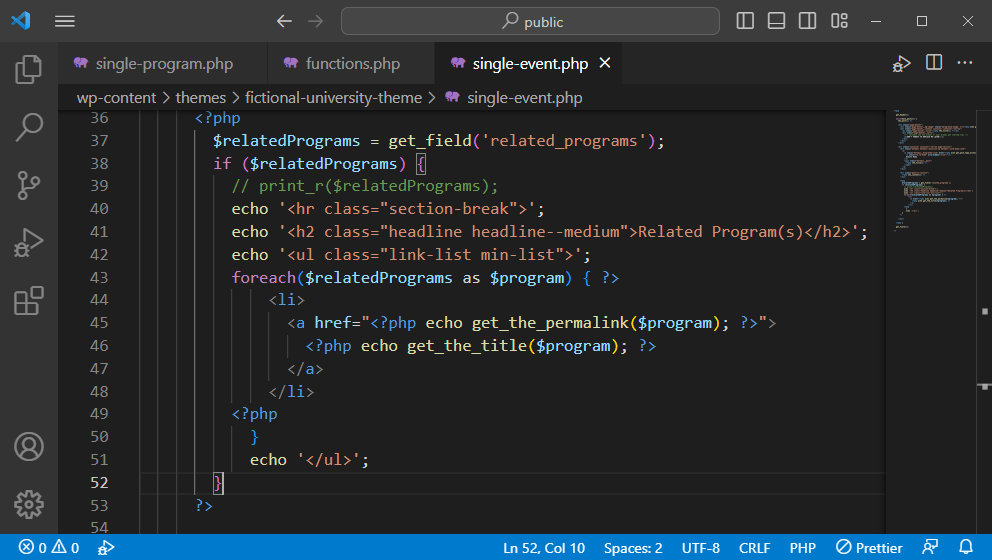
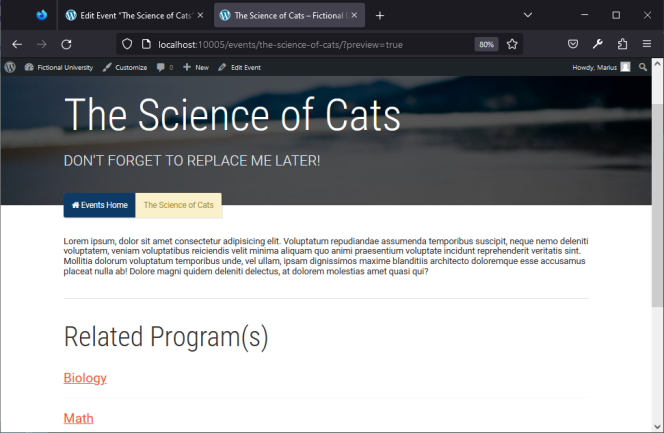
* În banda stângă a paginii de administrare va apărea item-ul Programs.
* Pe urmă, trebuie actualizată structura de permalink-uri în pagina de administrare.
* Postările din noul tip vor fi afişate iniţial de şablonul din fişierul single.php, de aceea se va crea în şablon dedicat pentru postări de acest tip în fişierul single-program.php.
* La fel, arhiva postărilor de noul tip va folosi şablonul generic de stil blog, archive.php, de aceea se va crea un şablon dedicat pentru afişarea tuturor postărilor de acest tip prin fişierul archive-program.php.
* Pentru ordonarea alfabetică crescătoarea a noilor postări după titlu lor, se adăugă o interogare personalizată dedicată pentru postările de tip program în cadrul acţiunii funcţiei add\_action() pentru evenimente pre\_get\_posts, interogare care va manipula interogarea implicită $query pentru afişarea tuturor tipurilor de postări de tip program.



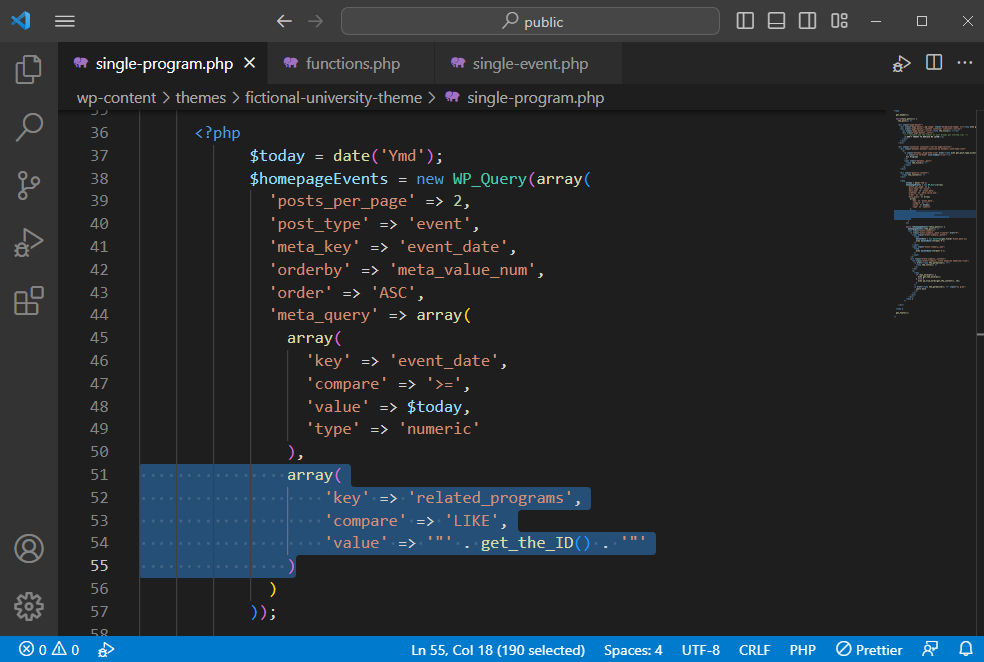
* Pentru crearea unei relaţii între postările de tip Event şi Program, se adaugă în câmp nou personalizat prin plugin-ul AFC, cu următoarele specificaţii:
  + Titlul grupului de câmpuri: Related Program
  + Butonul Add Field
    - Field Label: Related Program(s)
    - Field Name: related\_programs
    - Field Type: Relationship
    - Filter by Post Type: Program
    - Filters: Search
    - Return Format: Post Object
    - Location Rules: Show this field group if
      * Post Type is equal to Eventqq
      * La editarea unei postări de tip Event apare un câmp nou personalizat, Related Program(s), unde, prin clic pe un item din coloana din stânga, acesta este adăugat în dreapta.



* Preluarea câmpurilor postărilor de tip program asociate fiecărei postări de tip event se face prin funcţia get\_field(‘nume\_câmp’) dată de plugin-ul AFC şi se afişează valoarea acestor câmpuri în fişierul şablon al fiecărui eveniment, single-event.php.

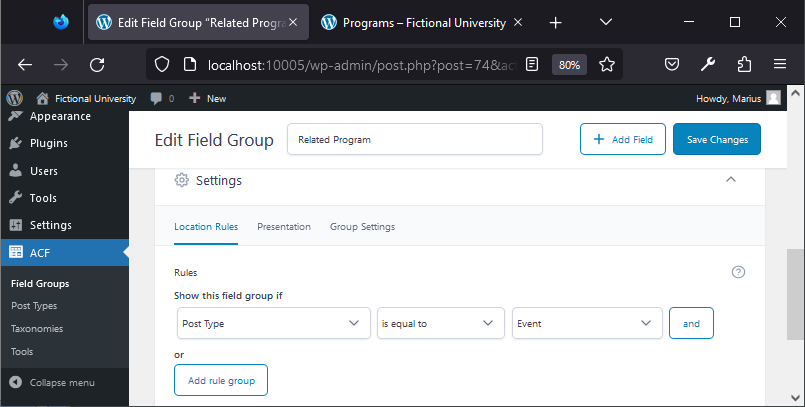
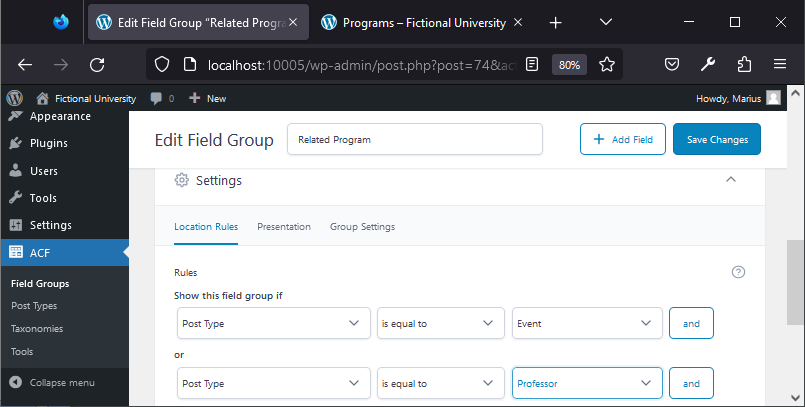
* Prin funcţia PHP print\_r(nume\_variabilă) se afişează datele despre acea variabilă stocate în baza de date.
* Anterior, postării de tip event The Science of Cats, i se atribuiseră două programe, Biology şi Math.
* Argumentul funcţiei WordPress get\_the\_title() este fie valoarea numerică a identificatorului unei anumite postări, fie un obiect postare WordPress.
* Relaţionarea inversă de la o postare de tip program la una de tip event se face printr-o interogare personalizată, meta\_query, suplimentară la baza de date, în fişierul şablon al unei postări de tip program –tabloul asociativ interior dintr-o interogare meta\_query poate fi asociat cu un filtru, dar în tabloul exterior meta\_query pot exista mai multe filtre.



* La comparaţie se foloseşte operatorul LIKE pentru conţine sau este ca, iar pentru valoare valoarea numerică a identificatorului este convertită în text printr-o pereche de ghilimele, prin concatenare.
* Când WordPress salvează un tablou într-o bază de date, el serializează datele, adică numerele sunt încadrate între ghilimele, de aceea numerele unui tablou trebuie căutate între ghilimele în baza de date.

## Tipul de postare Professor

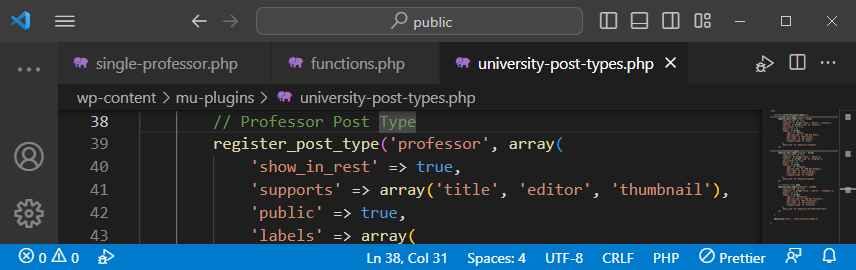
* După ce se creează postarea de tip professor, se face asocierea cu un program prin editarea câmpului personalizat Related Program prin adăugarea unui nou grup prin butonul Add rule group.

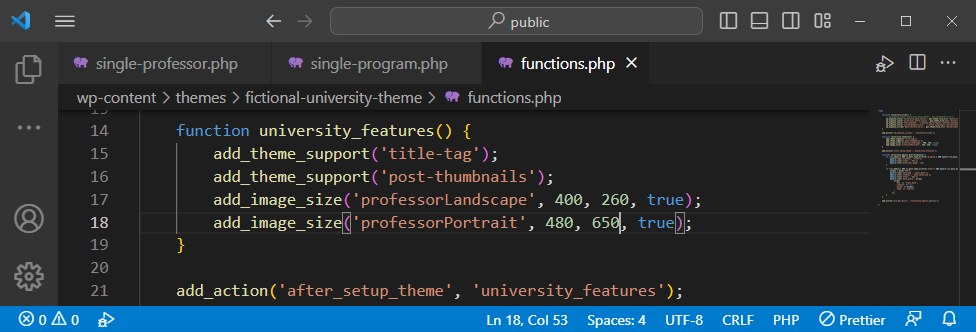
* Funcţia wp\_reset\_postdata() resetează obiectul global Post la valorile implicite de interogare, deci este indicat după fiecare interogare personalizată, în cazul în care sunt mai multe pe aceeaşi pagină.

## Imagini caracteristice

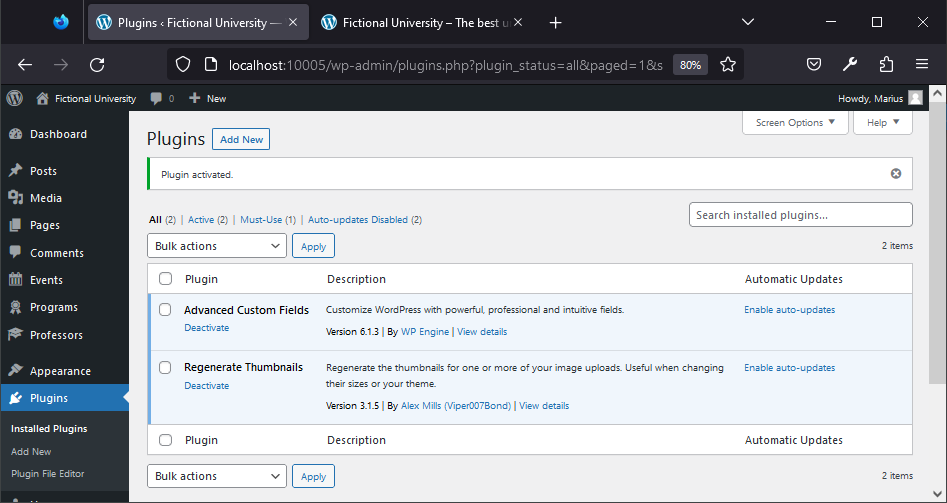
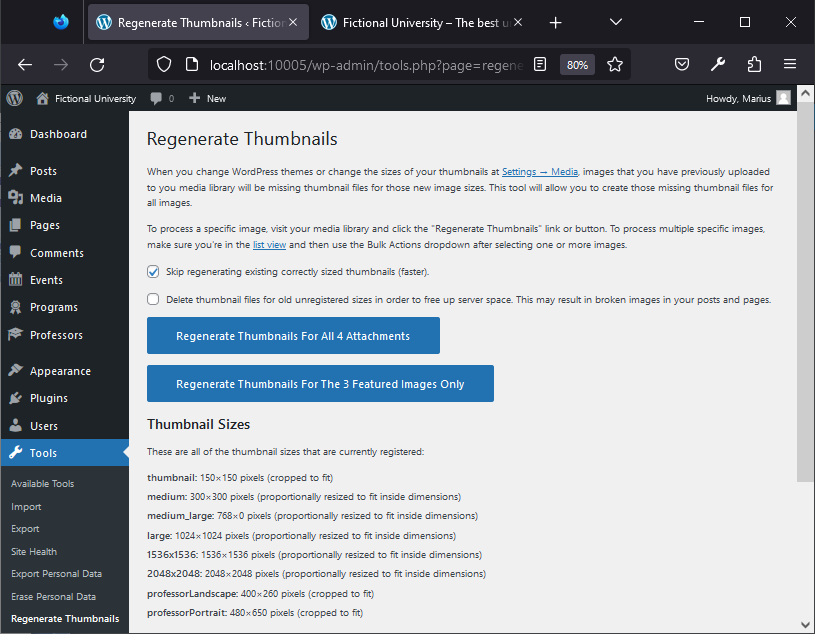
* Implicit, temele nu suportă caracteristica de imagini caracteristice (Featured Images) – se pot activa în funcţia functions.php cu add\_theme\_support(‘post-thumbnails’).
* Această setare are efect pentru postări de tip blog, dar pentru postări personalizate mai trebuie adăugat argumentul thumbnail în tabloul asociativ support de la funcţia regiter\_post\_type() pentru tipul de postare respectiv.



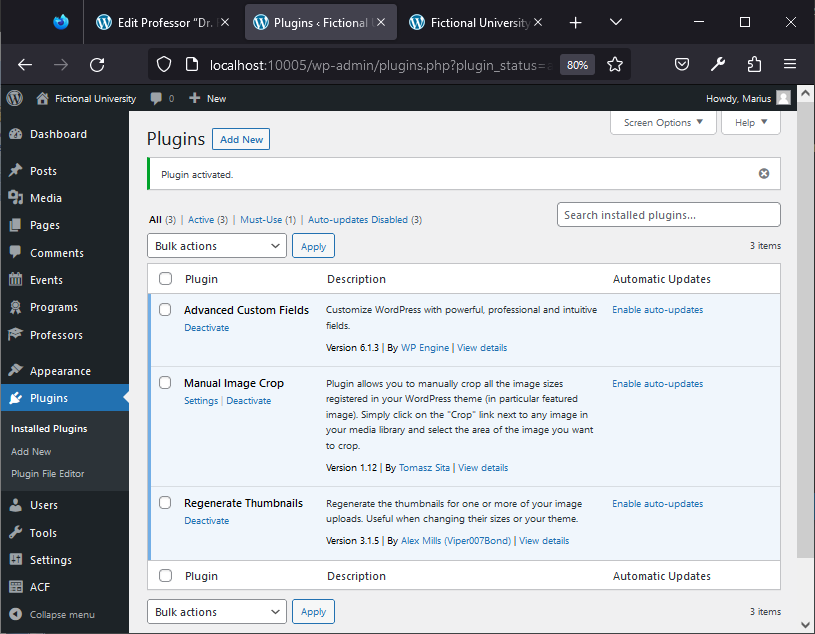
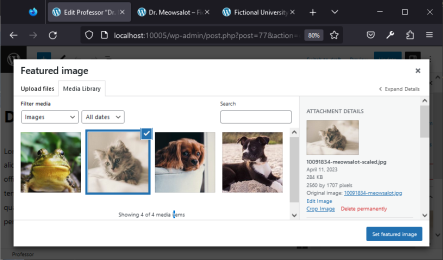
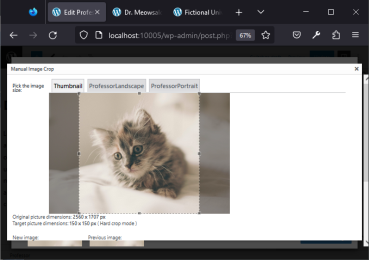
* Afişarea unei imagini asociate caracteristice cu o postare se face cu funcţia the\_post\_thumbnail(), iar adresa ei este dată d funcţia the\_post\_thumbnail\_url().
* Atunci când se încarcă o imagine într-o postare, WordPress generează automat copii ale acelei imagini la rezoluţii mai mici şi una pe orientare opusă faţă de cea originală (portrait/landscape).
* În plus, se poate specifica şi să se genereze o imagine la alte dimensiuni prin funcţia add\_image\_size(‘denumire\_format\_imagine’, ‘lăţime’, ‘înălţime’, true sau false sau array(‘left|right|center’, ‘top|bottom|center’) – true sau false sunt dacă se doreşte ca imaginea fie decupată sau nu, pentru a avea exact dimensiunile specificate în pixeli, iar tabloul asociativ determină colţul de la care se doreşte decuparea.



* După definirea altor dimensiuni pentru imagini, atunci când se încarcă o imagine nouă, se generează şi imagini la dimensiunile specificate.
* Crearea retroactivă de imagini la dimensiunile setate ulterior încărcării lor se poate face instalând şi activând plugin-ul Regenerate Thumbnails.

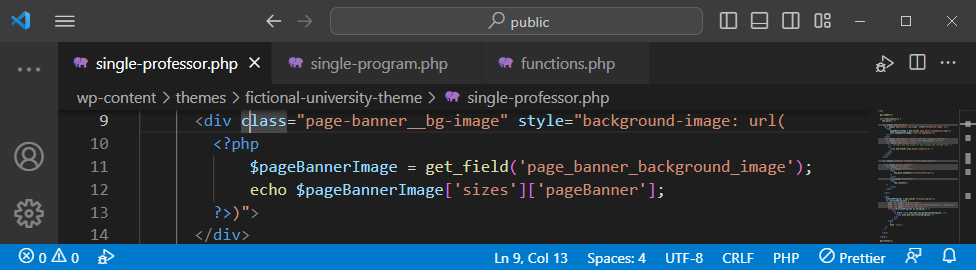
* În banda stângă a paginii de administrare este generată un meniu nou, Tools.
* Folosirea imaginilor cu dimensiunile stabilite în fişierul functions.php > function university\_features(), se face prin apelul funcţiei the\_post\_thumbnail() şi the\_post\_thumbnail\_url() cu argumentul denumirii formatului imaginii.
* Decuparea imaginilor se poate face şi cu ajutorul plugin-ului Manual Image Crop tomasz.

* Shift + F5: reîmprospătarea hard a browser-ului.

## Banner de pagină dinamic

* În WordPress, o postare poate să aibă numai o imagine caracteristică.
* Pentru a permite utilizatorilor să încarce un fişier sau un text, se creează câmpuri personalizate, care să permită aceste caracteristici.
* Se creează un grup de câmp, Page Banner, cu două câmpuri, unul pentru încărcarea textului şi unul pentru încărcarea imaginii.
* Afişarea subtitlului se face cu funcţia the\_field(‘nume\_subtitlu’).
* Pentru afişarea imaginii, întâi se preia într-o variabilă tabloul asociativ returnat de funcţia get\_field(‘nume\_imagine\_din\_postare’), apoi se trimite browser-ului rezultatul elementului din tabloul asociativ $nume\_variabilă[‘url’] sau, dacă s-a creat în prealabil o dimensiune personalizată de imagine, $nume\_variabilă[‘sizes’][‘nume\_dimensiune\_personalizată’].



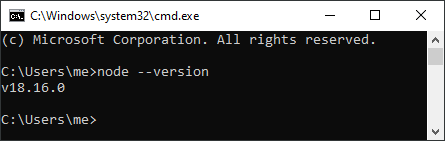
* WordPress decupează o imagine implicit începând din centrul imaginii.

## Reducerea codului duplicat – funcţia get\_template\_part()

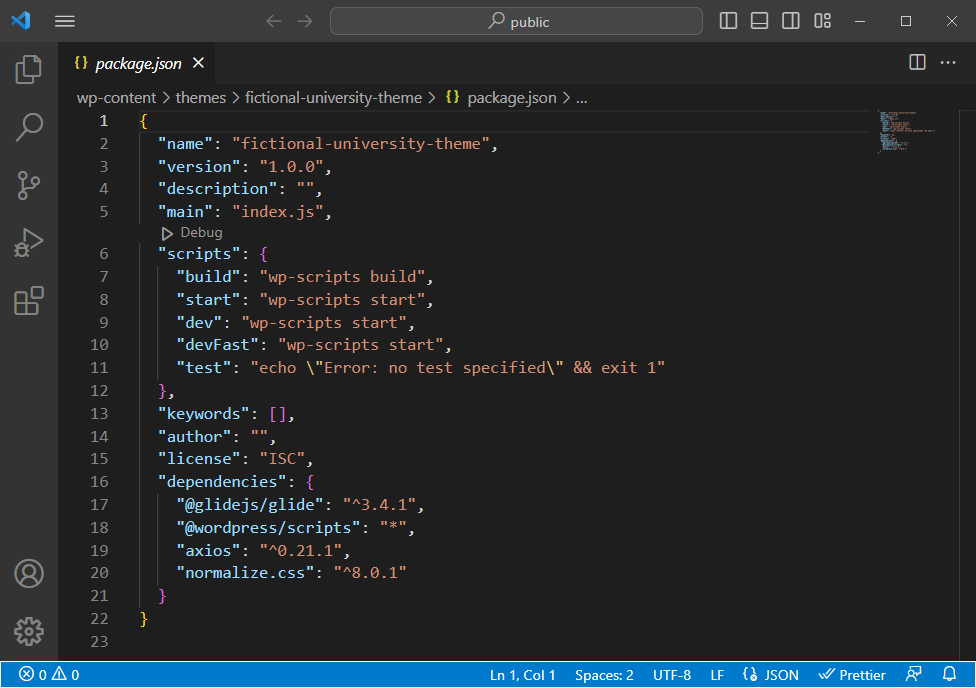
* Funcţia get\_template\_part() poate avea două argumente, primul, obligatoriu, fiind numele fişierului care va fi inclus, fără extensia .php, iar al doilea, poate fi o prelungire a numelui fişierului, unite prin semnul minus (get\_template\_part(‘alfa/beta’, ‘gama’) determină includerea conţinutului fişierului beta-gama.php din folder-ul alfa).
* Pentru cod flexibil, care acceptă parametri, se recomandă crearea unei funcţii adecvate în fişierul functions.php, iar pentru cod invariabil, se recomandă folosirea funcţiei get\_template\_parts().

## Prepararea calculatorului şi proiectului pentru JavaScript

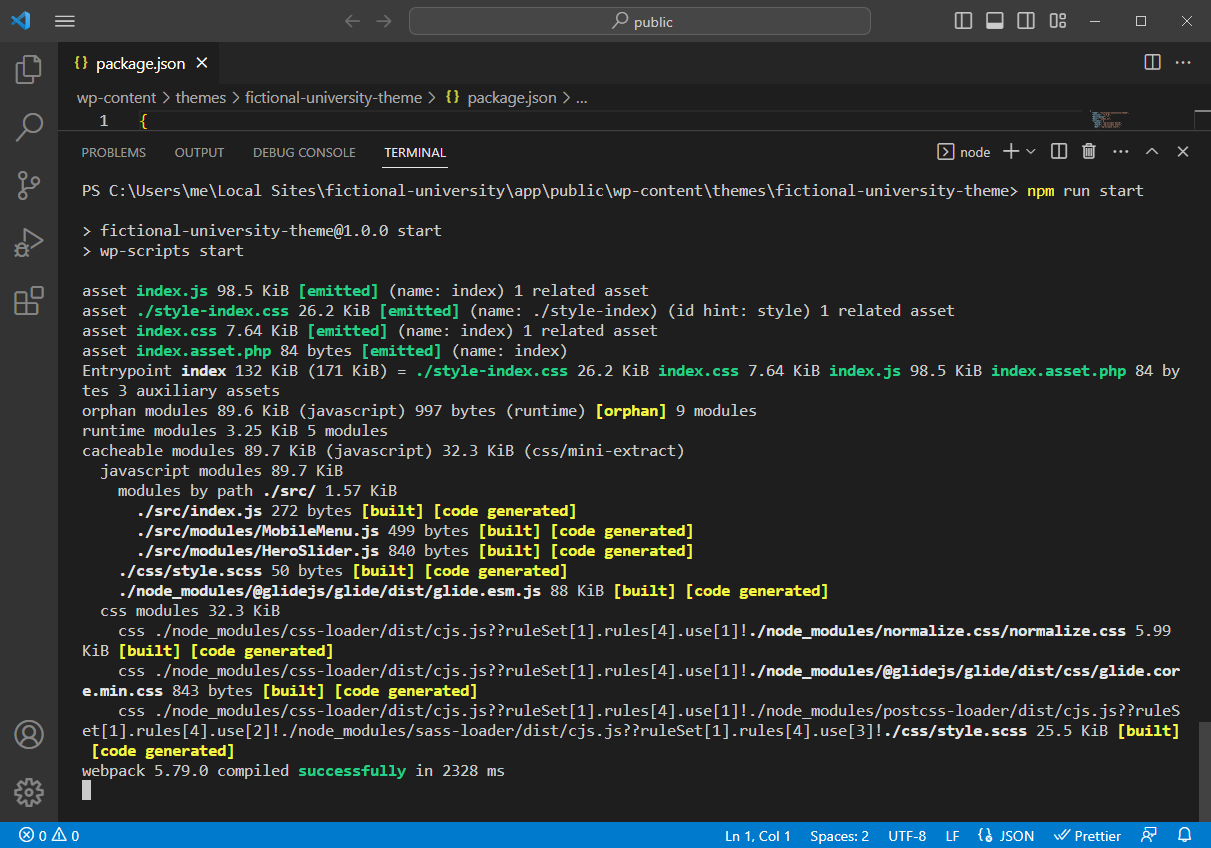
* Instrumentul oficial WordPress pentru editarea JavaScript este dat de pachetul @wordpress/script, care are nevoie şi de instalarea Node.js.
* Verificarea instalării Node.js în linia de comandă:



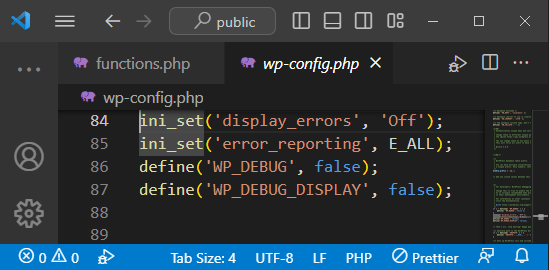
* Fişierul package.json conţine instrucțiunile pentru proiectul Node sau JavaScript – el trebuie să existe în folder-ul temei.



* Fişierul package.json conţine instrucțiunile pentru proiectul Node sau JavaScript – el trebuie să existe în folder-ul rădăcină al temei şi conţine mai multe dependenţe, care pot fi instalate din terminalul VS Code prin comanda npm install.
* După instalare, se poate folosi pachetul oficial WordPress pentru JavaScript, @wordpress/script.
* În fişierul package.json sunt referite şi două script-uri, build şi start.
* Task-ul start se lansează în execuţie prin linia de comandă prin npm run start şi continuă să ruleze în fundal până când i se spune explicit să se oprească.

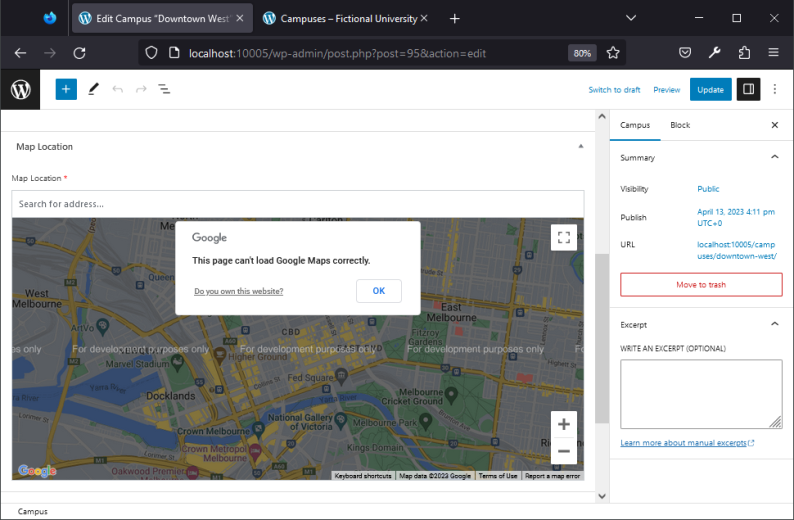


* El urmăreşte, în fundal, situaţiile când se salvează schimbări în fişiere.
* În folder-ul rădăcină al temei, trebuie să existe folder-ul src, iar în interiorul său trebuie să existe fişierul index.js, pentru că aceste două elemente sunt căutate de instrumentul pentru JavaScript al lui WordPress.
* Acest fişier index.js este codul sursă optimizat pentru a fi citit de utilizator, însă instrumentul JavaScript pentru WordPress împachetează tot conţinutul şi îl compilează într-un fişier final index.js, situat în folder-ul build din rădăcina temei – acest fişier final nu este într-un format uşor de urmărit şi, de aceea, el nu se editează direct niciodată, ci numai cel din folder-ul src.
* Pachetul oficial JavaScript pentru WordPress manipulează, de asemenea, şi fişierele CSS, SAS sau SCSS.
* Fişierele din folder-ul build sunt optimizate pentru browser-ele web, nu pentru utilizatori, deci nu se editează direct niciodată.
* Alt folder cu fişiere editabile direct este css, pentru fişierele CSS.
* Oprirea pachetului se face prin Ctrl + C la linia de comandă.
* Task-ul build nu rulează în fundal, el rulează o singură dată – ia tot codul sursă din folder-ul src şi css şi împachetează totul în folder-ul build.
* Pentru importul unui site trebuie instalat şi activat plugin-ul all in one de ServMask, apoi trebuie actualizată structura de permalink-uri pentru noul site şi instalate modulele Node.js şi dependenţele din fişierul package.json prin npm install în terminal. Acest plugin poate fi folosit şi pentru exportul unui site.
* Anularea mesajelor de avertizare/eroare în WordPress se poate face prin editarea fişierului wp\_config.php al site-ului, situat în folder-ul app/public:

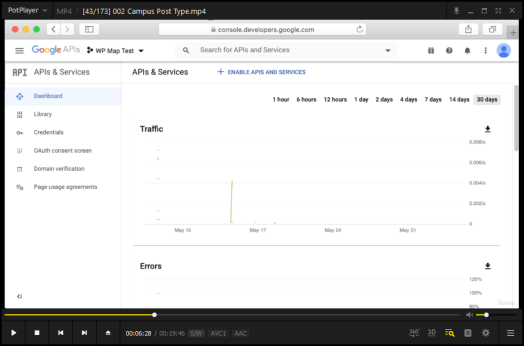
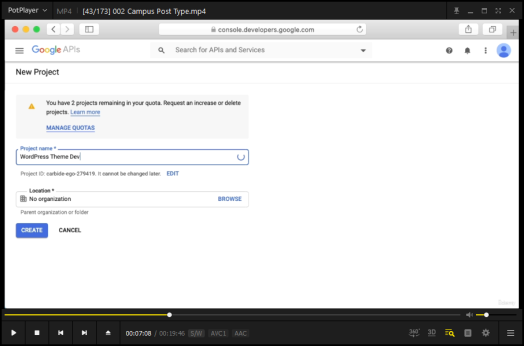
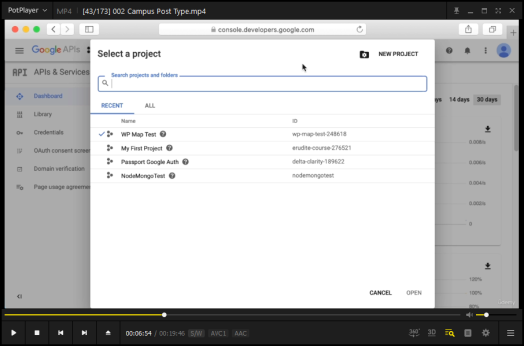


## Google Map

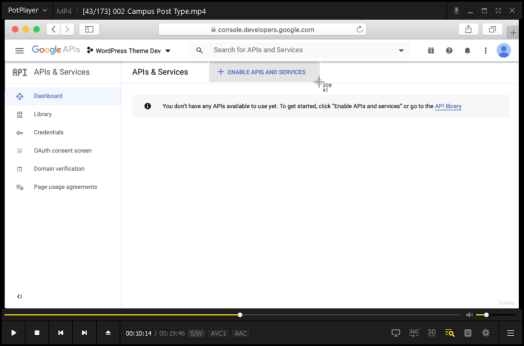
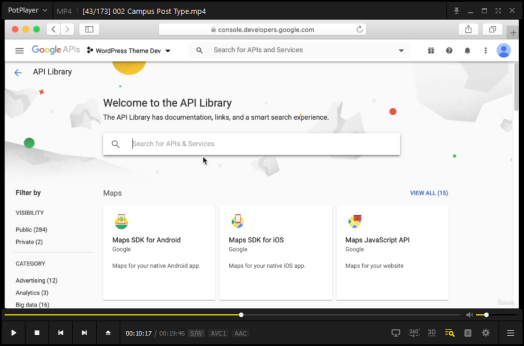
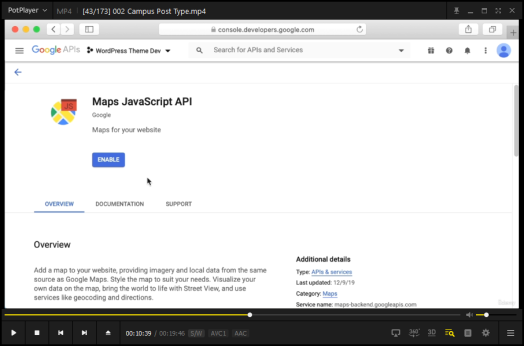
* Pentru a adăuga o hartă Google în site, în care să poată fi punctată o locaţie, se poate crea un tip de câmp personalizat prin ACF, de tip Google Map – de ex. Map Location, şi se setează afişarea acestui câmp personalizat Google Map numai pentru postări de tip personalizat campus.
* Iniţial, în pagina de administrare pentru un astfel de tip personalizat de postare, apare un mesaj de eroare în cadrul acestui câmp.

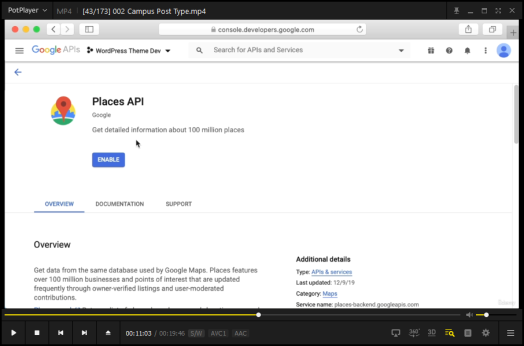
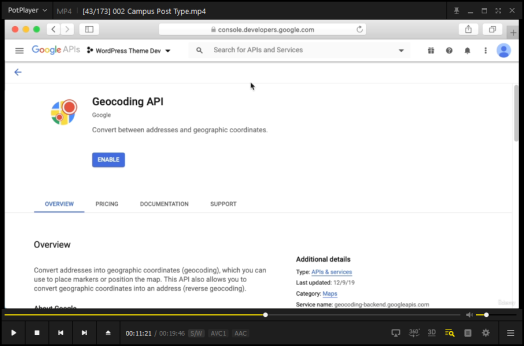


* Pentru a accesa serviciul Google Map într-un site web, este necesară cheia API pentru Google Maps, pentru a putea comunica cu serviciul Google Map utilizând JavaScript, în loc de a folosi un iframe cu adresa unei hărţi stocate pe serverul Google Maps.
* O cheie API este doar un şir de litere şi cifre, similar cu o parolă, doar că nu trebuie să fie ţinută secretă complet întotdeauna – de fapt, este o strângere secretă de mână care se face cu serverele Google, facpt care face posibilă comunicarea cu serviciul Google Maps.
* Un utilizator cu cont Google îşi poate crea o cheie API prin logarea la contul respectiv şi accesarea adresei <https://console.developers.google.com/>, crearea unui proiect cu orice nume şi selectarea acelui proiect.

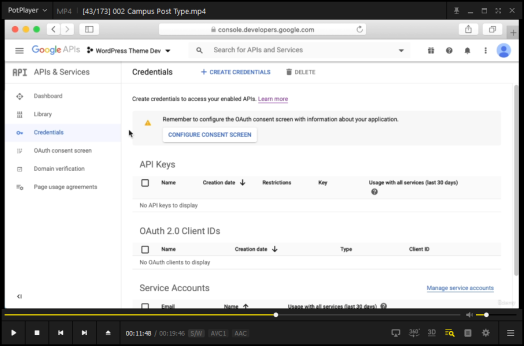
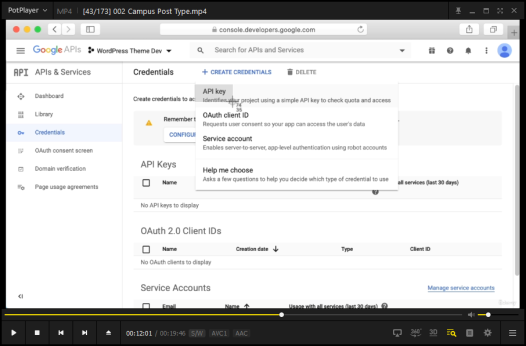
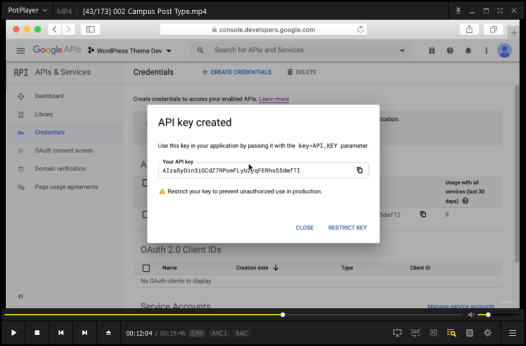
 

* Iniţial, utilizarea unei chei API era gratuită, dar acum trebuie activată facturarea în cadrul contului Google, furnizând date despre un card de credit şi adresa, deşi Google spune că primele 12 luni rămân gratuite sau dacă utilizatorul impune anumite limite de accesare cu cheile API.
* Odată selectat proiectul nou creat, trebuie selectat butonul ENABLE APIS AND SERVICES şi se caută şi se activează trei API-uri: Maps JavaScript API, Places API şi Geocoding API.

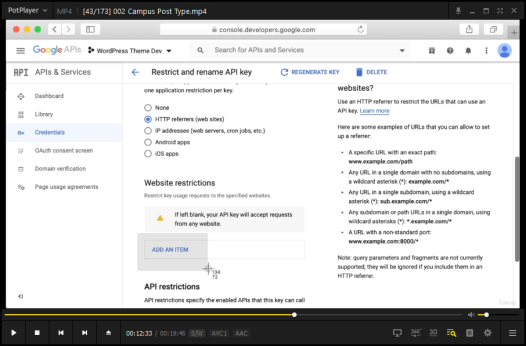
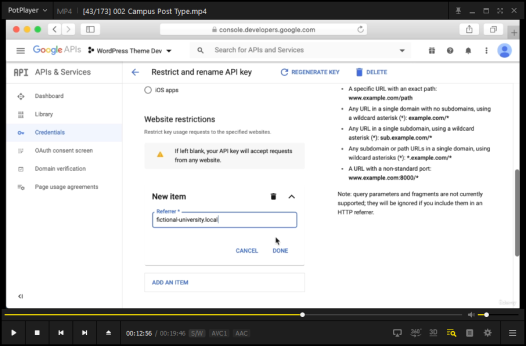
  

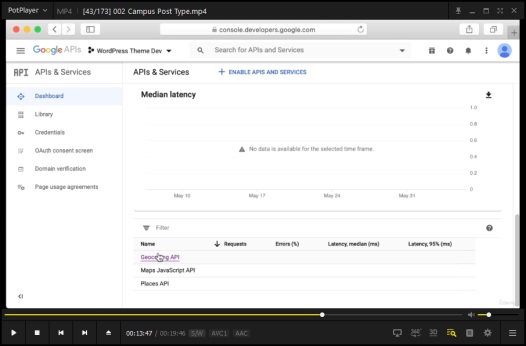
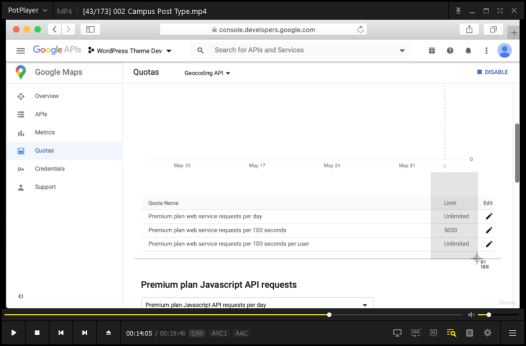
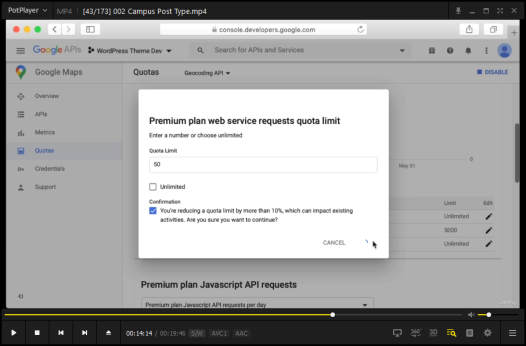
* După activarea celor trei API-uri şi selectarea proiectului, se poate trece la crearea cheii API prin opţiunea Credentials din banda stângă, selectarea butonului CREATE CREDENTIALS şi alegerea opţiunii API key.

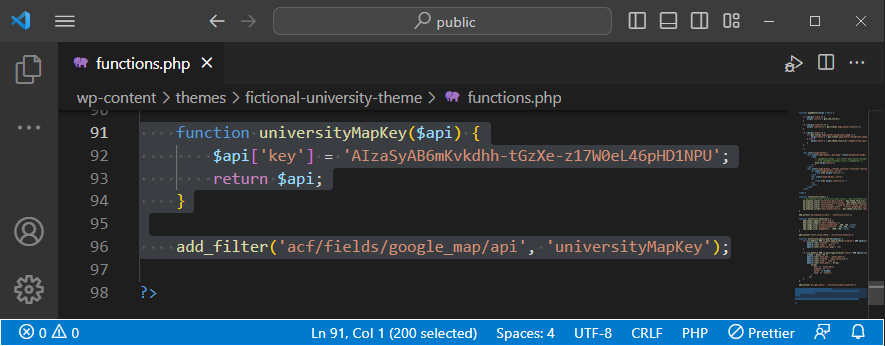
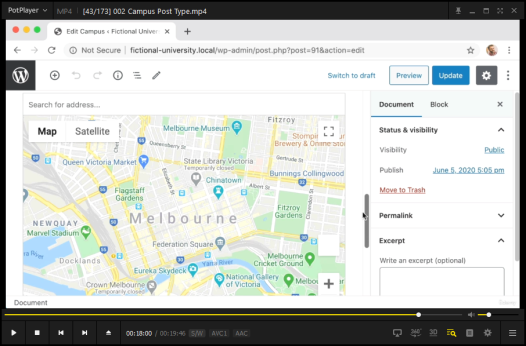
* După crearea cheii API se alege opţiunea RESTRICT KEY şi următoarele restricţii – la sfârşitul adresei website-ului de la New Item este bine să se adauge un \*, de ex. http://localhost:10005/\*

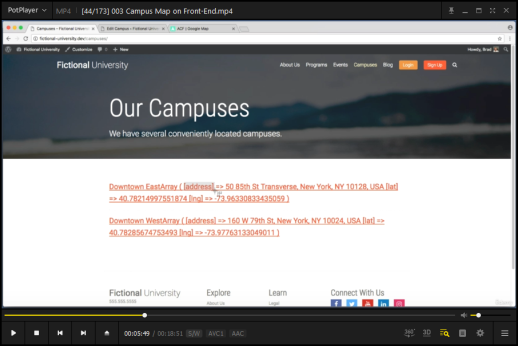
* Pentru a seta numărul maxim de reîmprospătări ale paginii, a.î. să nu se depăşească limita pentru care se facturează serviciul Google Map, se stabilesc aceste bareme pentru fiecare API.

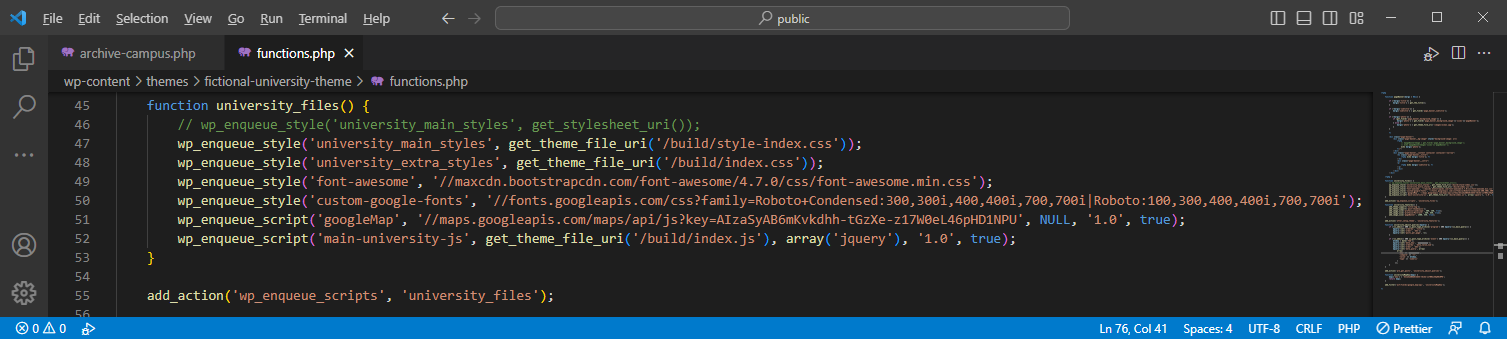
* La sfârşit, trebuie adăugate datele de facturare.
* După asocierea contului cu un cont de card bancar, se poate adăuga un filtru în fişierul functions.php, filtru care să acceseze plugin-ul ACF şi comunicarea existenţei unei chei API pentru Google Maps, iar acest filtru va comunica un argument $api funcţiei care va fi al doilea argument al filtrului, funcţie prin care se setează cheia acelui API, împrospătarea paginii de administrare a postării respective nemaiafişând mesajul de eroare.

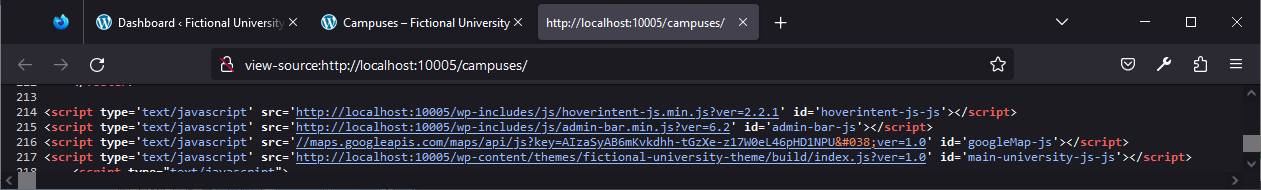


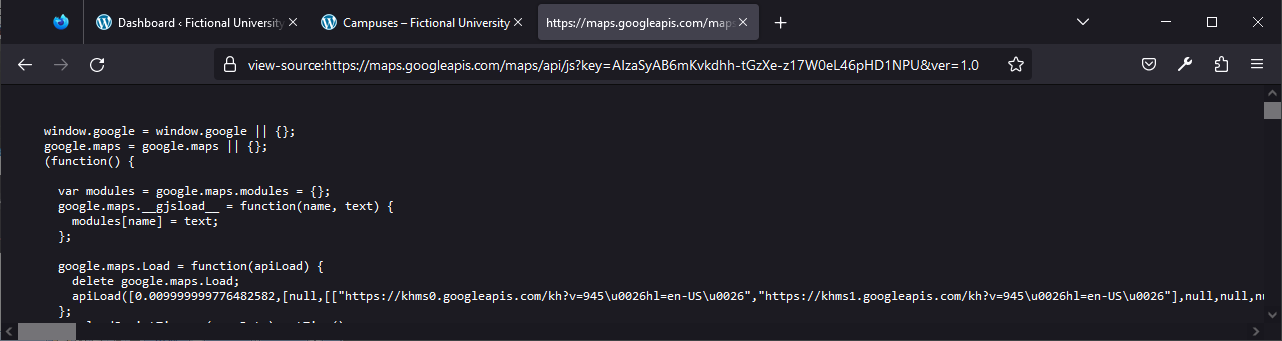
* În câmpul Search for address se poate stabili o adresă, iar apoi, printr-un clic, se poate puncta referinţa locului respectiv.
* Funcţia get\_field(‘nume\_locaţie\_hartă’) returnează un tablou asociativ cu valori pentru adresă, latitudine şi longitudine (address, lat şi lng).



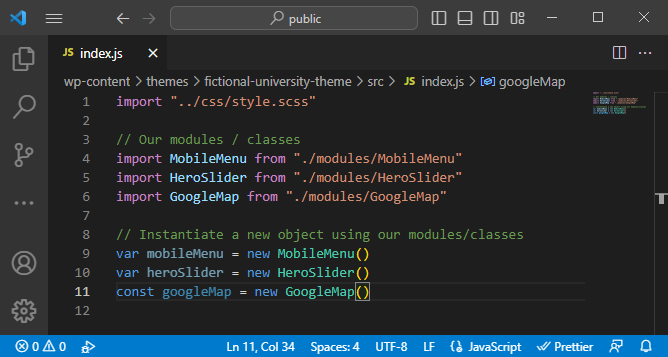
* Încărcarea fişierului JavaScript de pe serverul Google se face prin specificarea în fişierul functions.php, în funcţia corespunzătoare acţiunii wp\_enqueue\_script, a unei noi intrări pentru serverul Google.



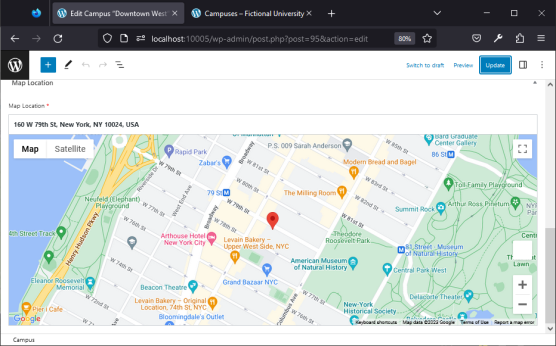
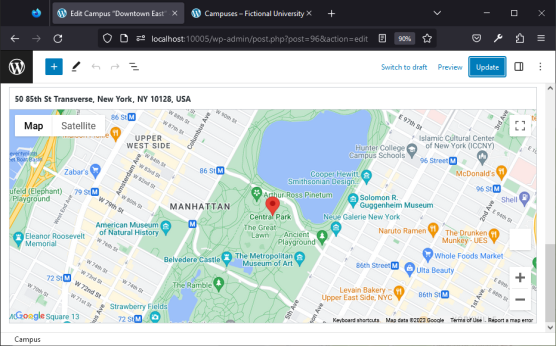




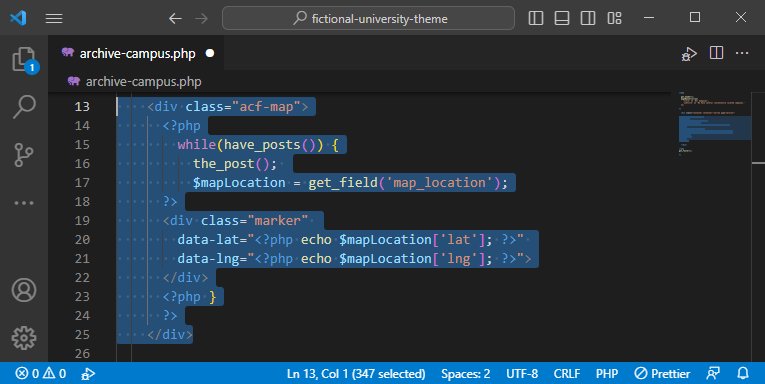
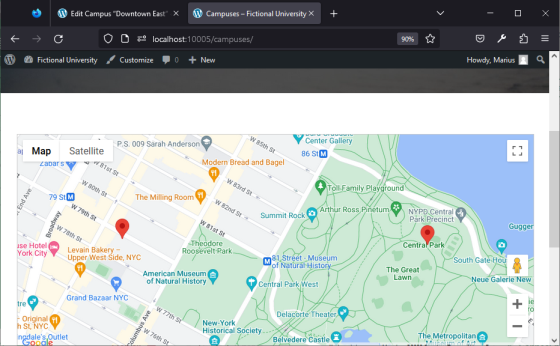
* Manipularea datelor primite de la serverul Google se poate face printr-un fişier JavaScript, pornind de la documentaţia de la adresa <https://www.advancedcustomfields.com/resources/google-map/>.
* În final, se va folosi fişierul src/modules/GoogleMap.js din tema site-ului, importat în fişierul principal JavaScript, src/index.js.



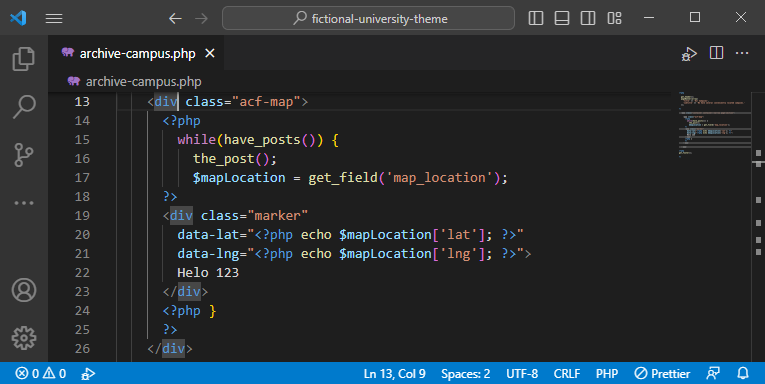
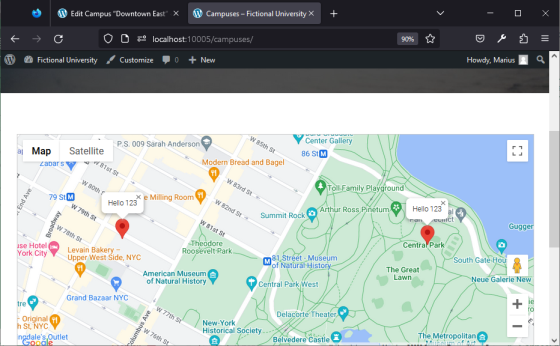
* Setarea locaţiei hărţii pentru fiecare postare personalizată care are un câmp personalizat ACF de tip Map Location se face în pagina de administrare pentru postarea respectivă.

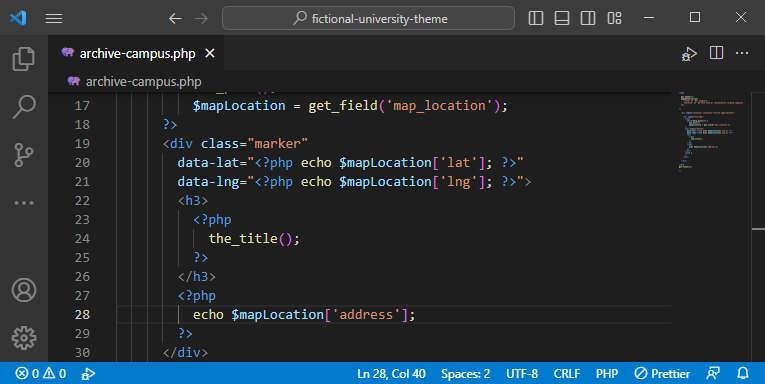
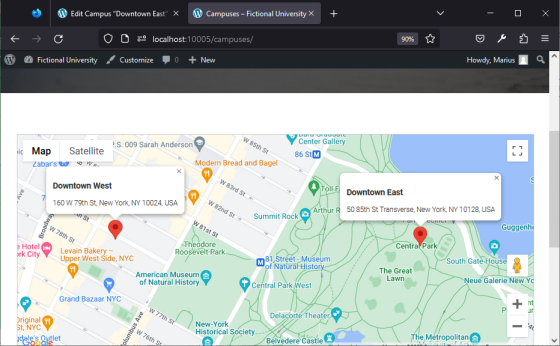
* Afişarea hărţii Google Map se face într-un element div de clasă marker, cu atributele data-lat şi data-lng, situat într-un alt element div de clasă acf-map.

* Dacă în interiorul elementului div de clasă marker se scrie un text, acest text va fi afişat la un clic pe fiecare marker de locaţie din fiecare postare.

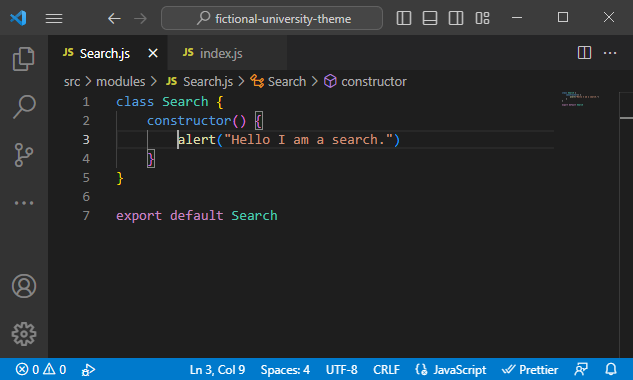
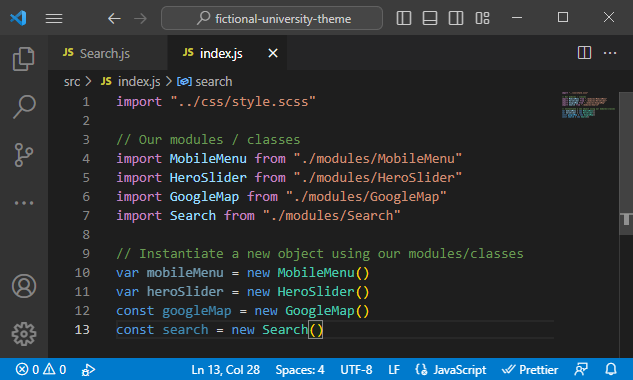
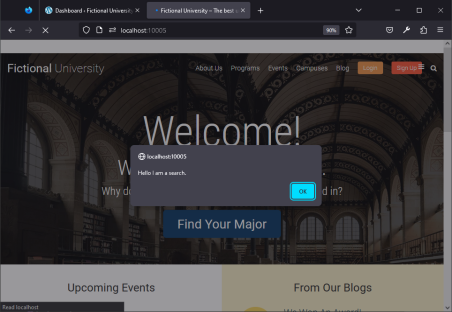
 

* Dar conţinutul acestui element poate fi folosit pentru a afişa titlu postării şi adresa locaţiei.

## Rezultatele căutării în direct (cu JavaScript)

* JavaScript permite manipularea paginii în timp real.
* În folderul src/modules al temei curente, se creează fişierul Search.js, iar în el se defineşte o clasă Search, care va fi importată în fişierul principal index.js.

* Codul care există în interiorul unei clase nu este propriu-zis executat până când nu este creat un obiect care utilizează clasa respectivă ca schemă de planificare.
* Corpul unei clase poate fi împărţit în trei secţiuni:
  + Descriere şi creare/iniţializare obiect.
  + Evenimente.
  + Metode (funcţii, acţiuni).
* Pentru utilizarea bibliotecii jQuery, trebuie folosită la începutul fişierului declaraţia import $ from ‘jquery’;.
* Selectarea elementelor HTML cu jQuery se face cu $(".denumire\_class") sau cu $("#denumire\_id").