Contents]

[Fatah Gabrial „WordPress Course - Beginners Guide to WordPress 5 (2022)” [Udemy] (2022) 2](#_Toc131521078)

[Brad Schiff „Become a WordPress Developer.. Unlocking Power With Code” [Udemy] (2022) 4](#_Toc131521079)

[PHP 6](#_Toc131521080)

[Teme (Themes) 7](#_Toc131521081)

[Postări (Posts) 8](#_Toc131521082)

[Antete şi subsoluri (Headers and Footers) 9](#_Toc131521083)

[Fişierul functions.php 9](#_Toc131521084)

[Pagini (Pages) 9](#_Toc131521085)

[Recomandări 10](#_Toc131521086)

[Meniuri de navigare dinamice 10](#_Toc131521087)

[Aspectul de blog al site-ului web 11](#_Toc131521088)

[Ecrane de arhive 11](#_Toc131521089)

[Interogări personalizate 12](#_Toc131521090)

# 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

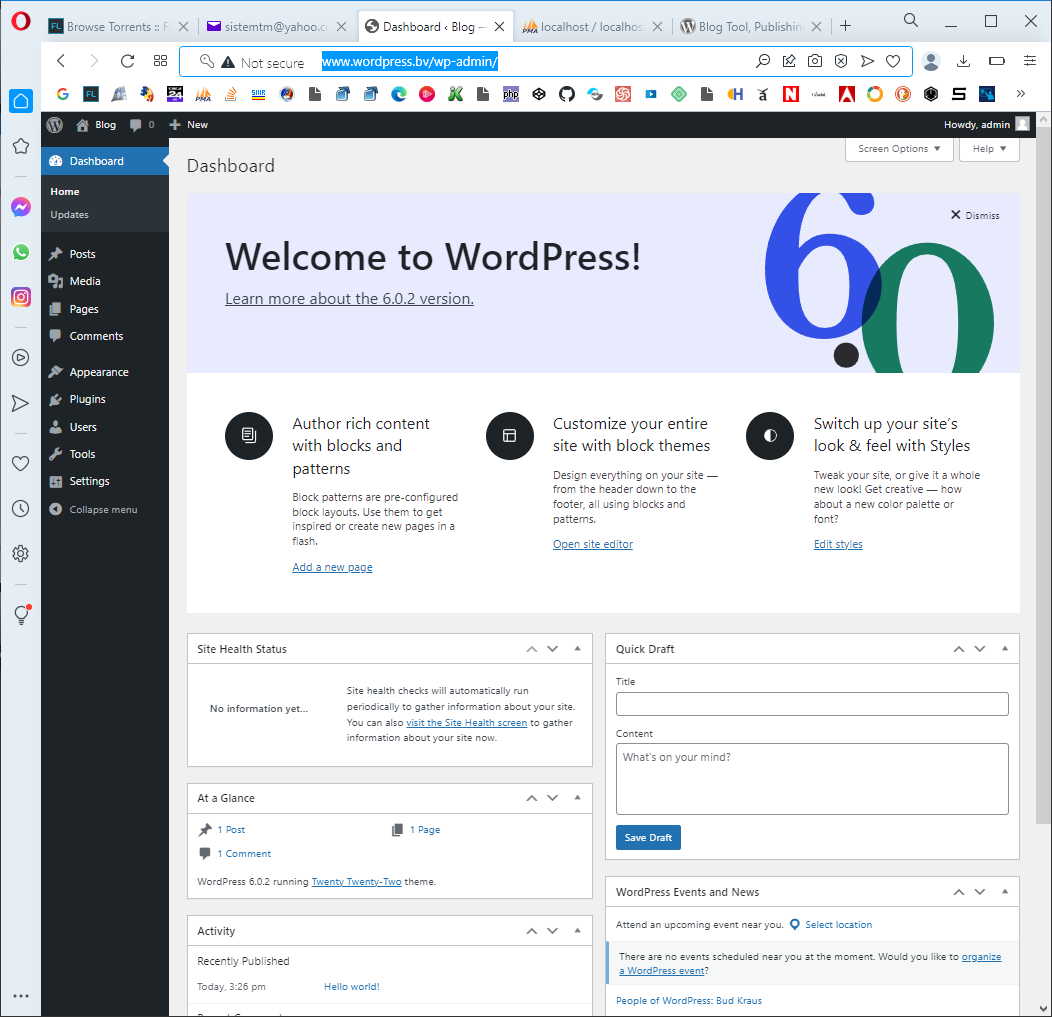
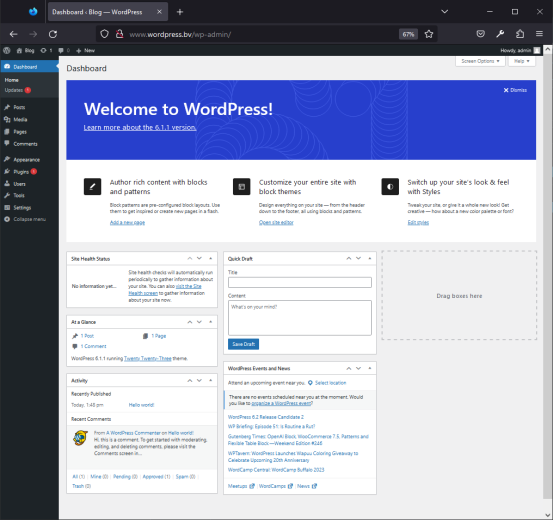
* + Success

Log in

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

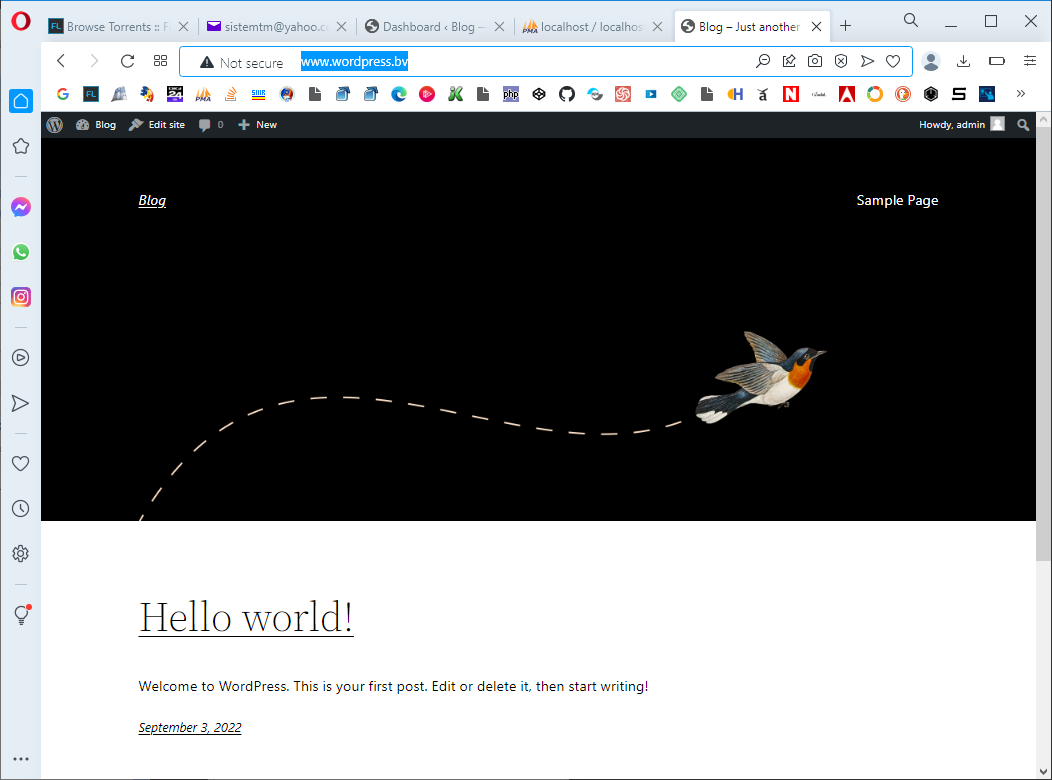
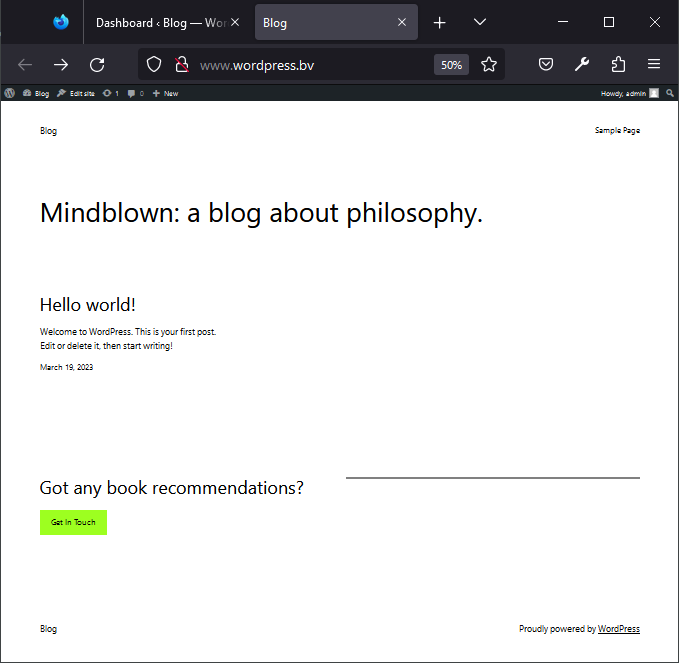
Log in

* <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

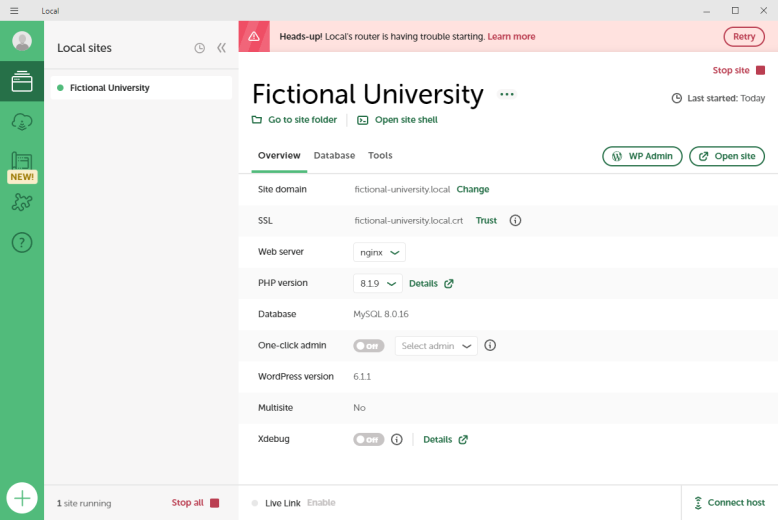
WordPress Email => [dev-email@wpengine.local](mailto:dev-email@wpengine.local)

Windows Defender Firewall has blocked some features of this app

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

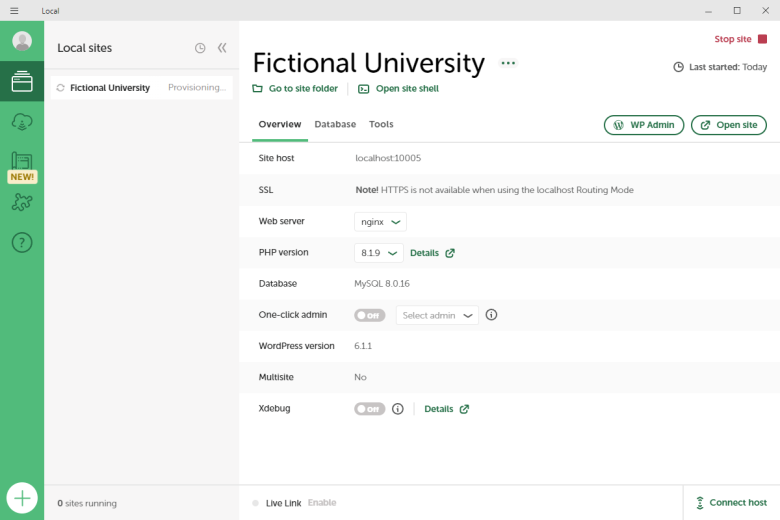
Allow access

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

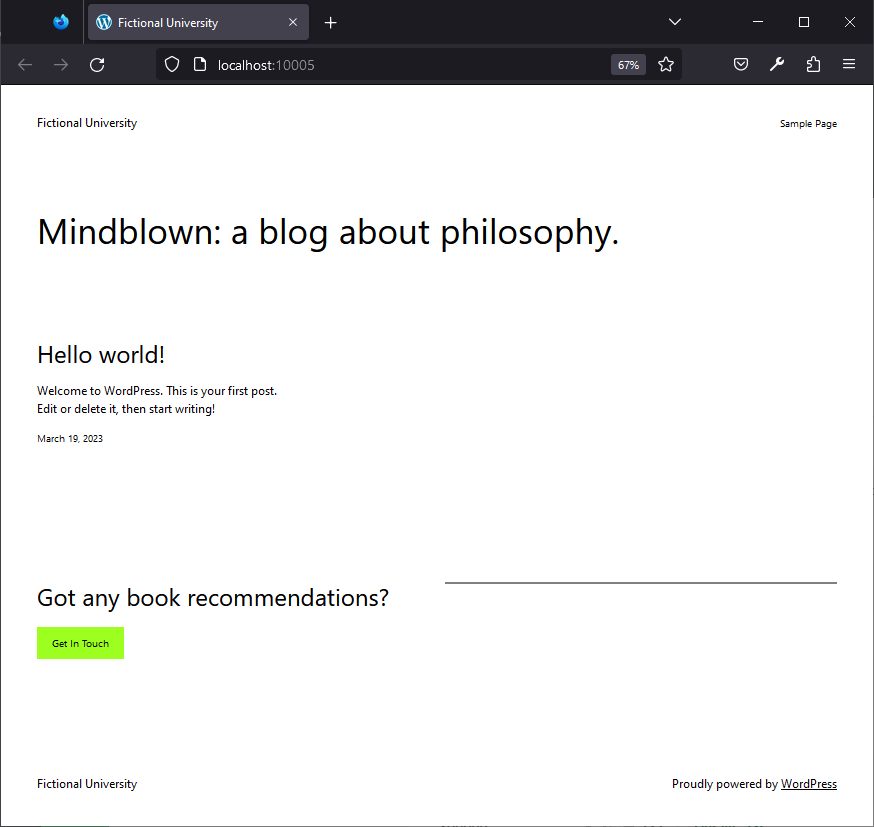


* + - Setarea ≡ > Preferences > Advanced > Router Mode >localhost

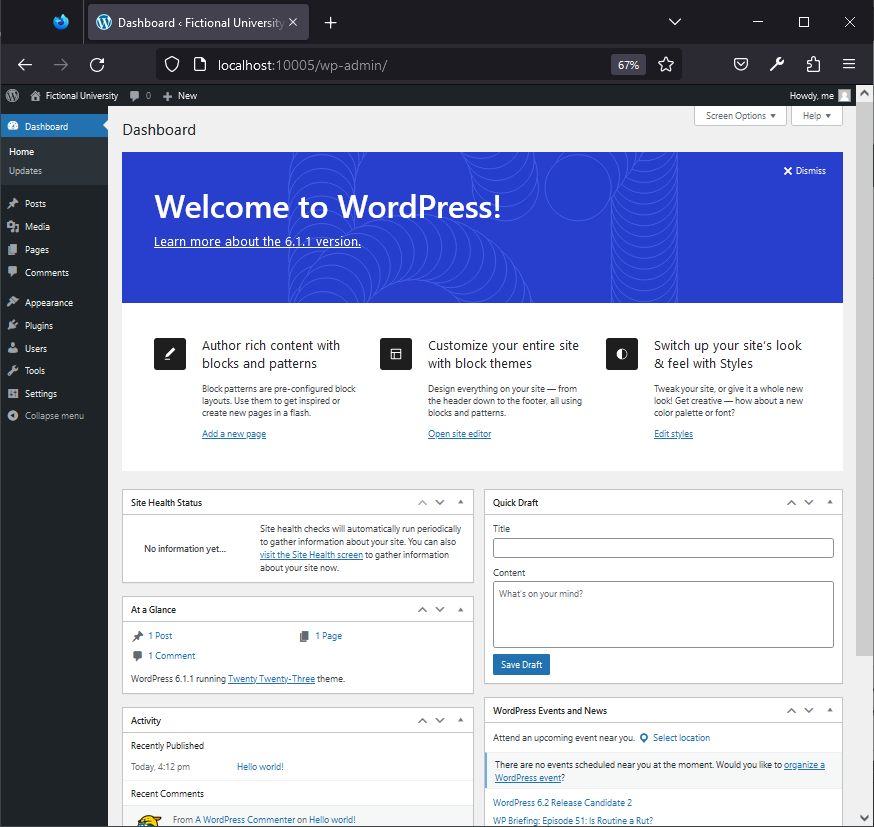
=> Dispare mesajul de eroare legat de router-ul local



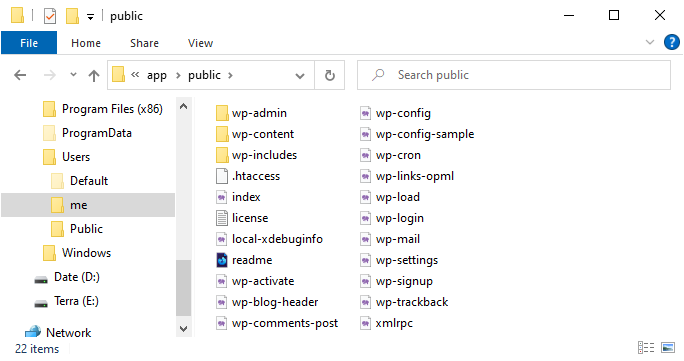
* + - Site-urile nou create vor fi localizate în folder-ul C:\Users\me\Local Sites – locul se poate schimba din ≡ > Preferences > New site defaults > Sites path => E:\work\vhosts\www.local.bv\public\_html
    - Vizualizarea site-ului: butonul Open Site



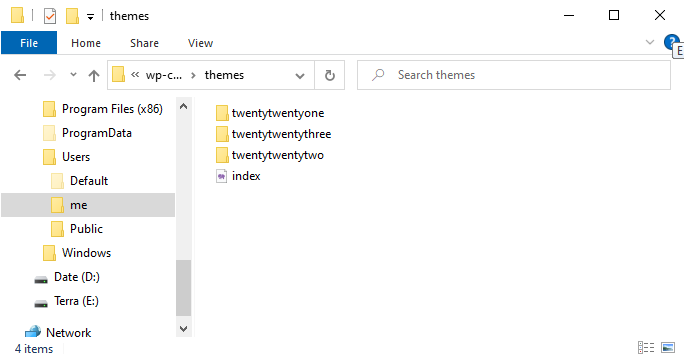
* Trecerea în browser în modul de administrare: <http://localhost:10005/wp-admin/>



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



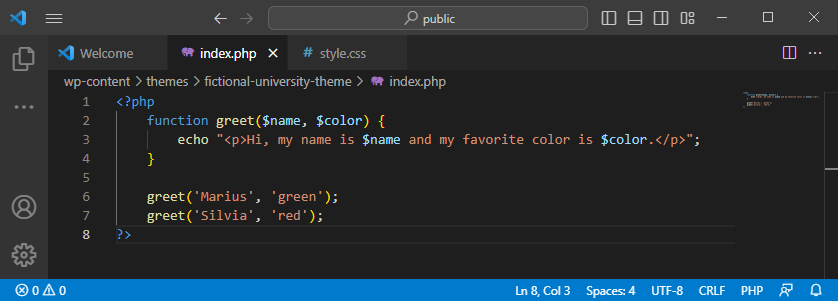
* În folder-ul C:\Users\me\Local Sites\fictional-university\app\public\wp-content\themes se află temele instalate



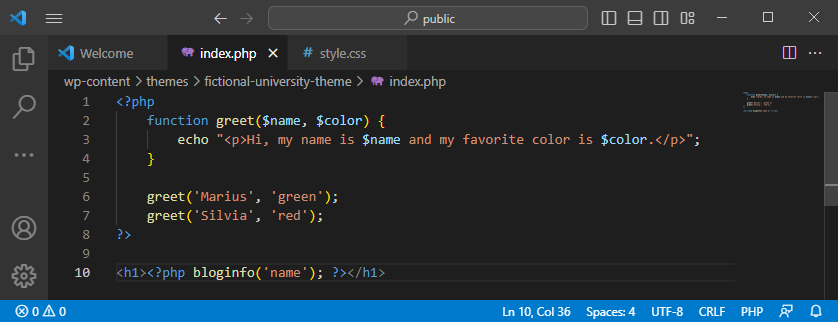
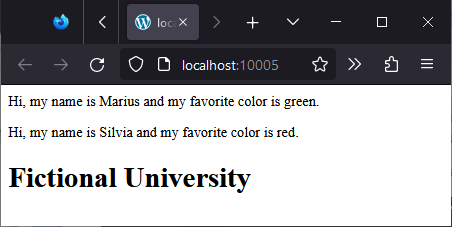
* Deoarece numele utilizatorului nu poate fi schimbat, pentru a afişa alt nume într-o pagină prin cod PHP, în pagina de administrare a site-ului, la secţiunea Users > me > Nickname se poate specifica numele respective, iar la Display name publicly as se poate alege ce nume să se afişeze.

## 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).

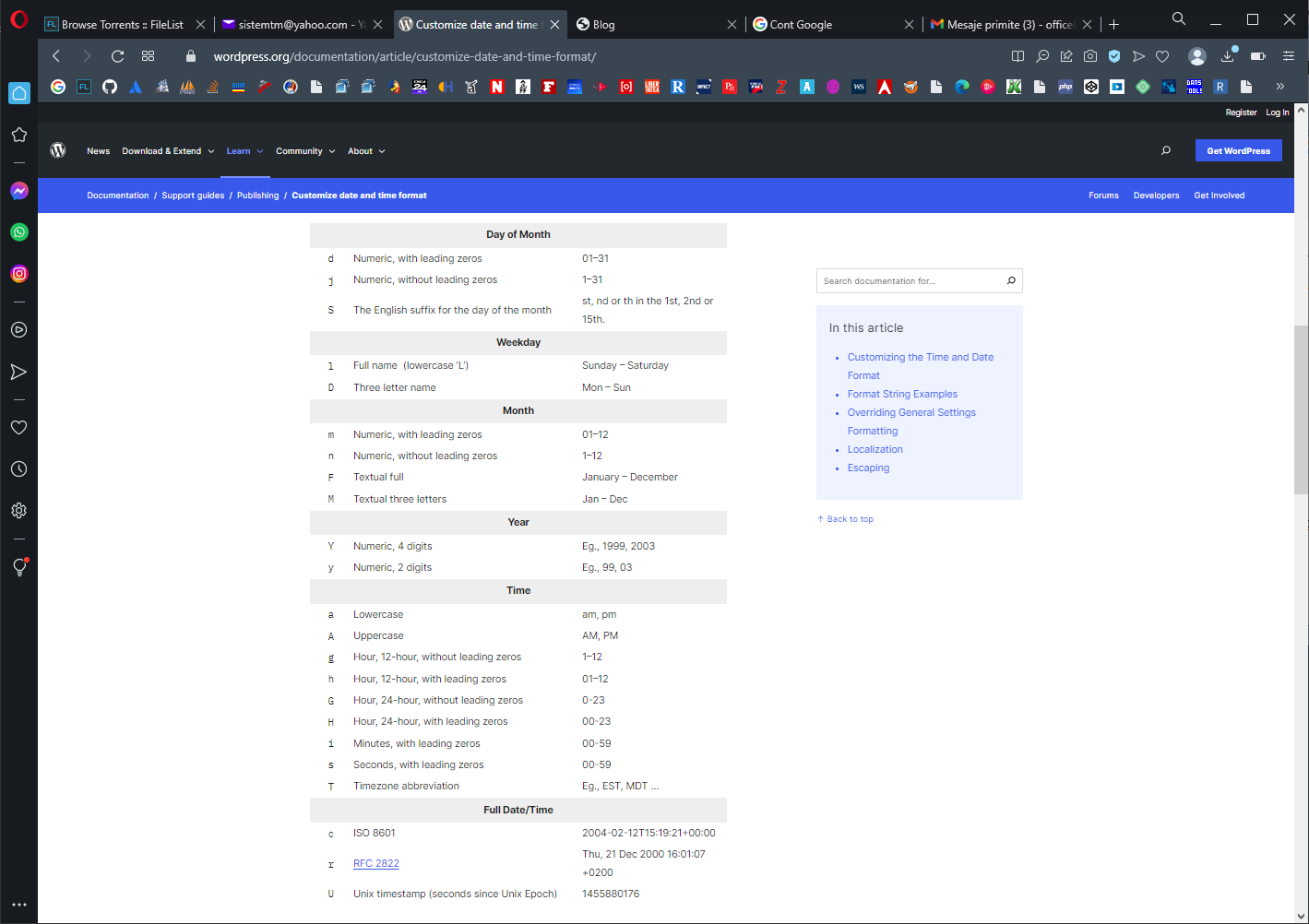
$nume\_variabilă = array(‘item1’, ‘item2’, …);

* 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/> :

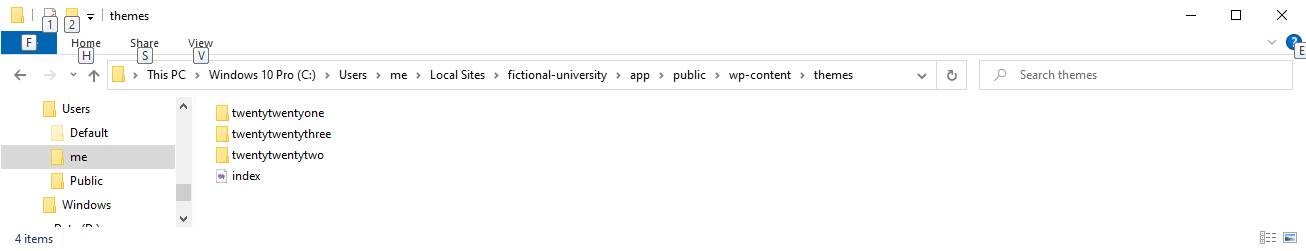
the\_time(‘d.m.Y’) => 25.03.2023.



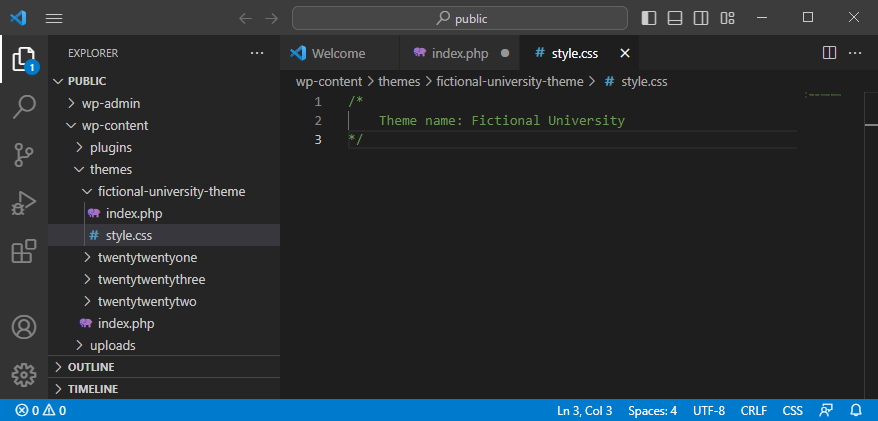
* Funcţia site\_title() returnează adresa site-ului, iar dacă i se furnizează ca argument ‘/valoare\_slug’ returnează link-ul paginii cu slug-ul respectiv.
* Funcţia wp\_trim\_words(textul\_de\_trunchiat, limita\_de\_cuvinte) trunchiază un text la un număr specificat de cuvinte.
* Funcţia get\_post\_type() 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)

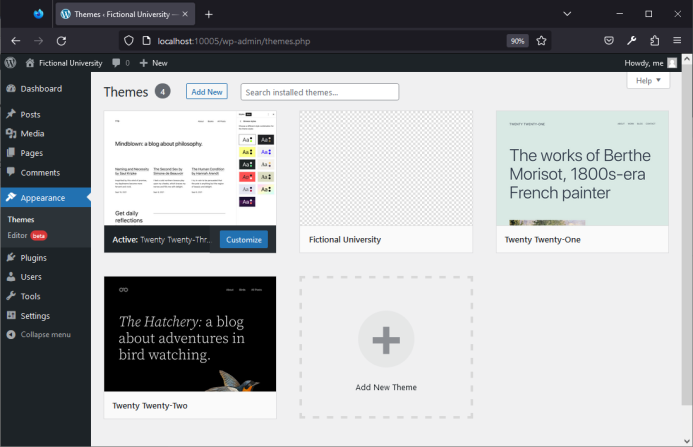
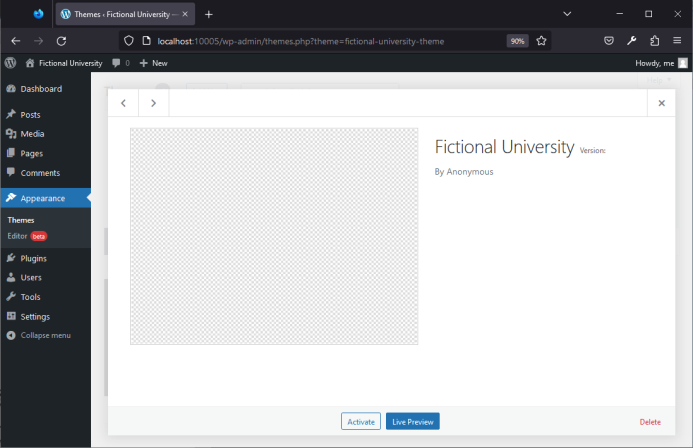
* Locație: C:\Users\me\Local Sites\fictional-university\app\public\wp-content\themes, adică în folder-ul „app\public\wp-content\themes” al site-ului.



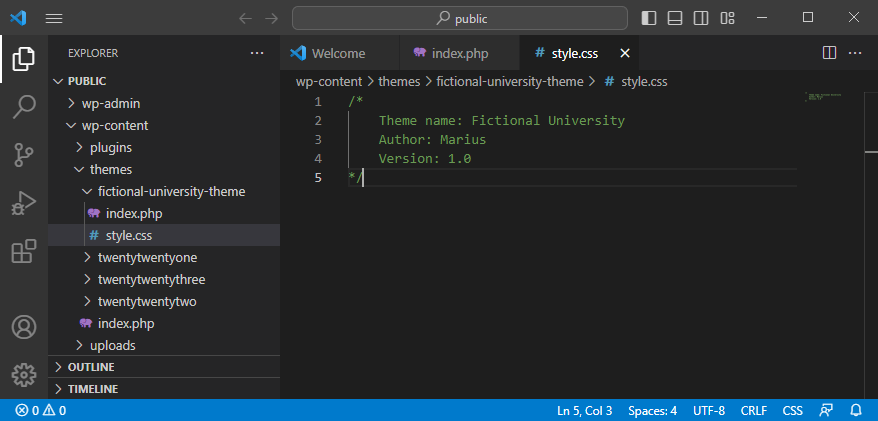
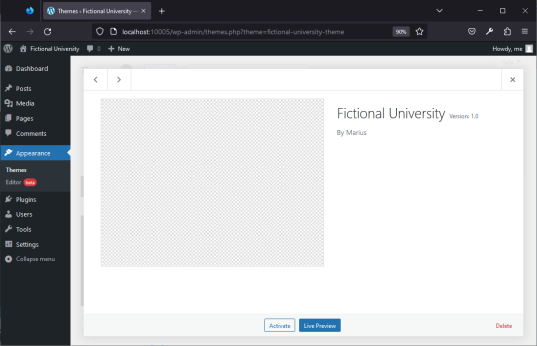
* Crearea unei teme noi: crearea unui folder nou, care să cuprindă ce puţin două fişiere, index.php şi style.css.
* Fişierul style.css mai are rolul de a furniza informaţii despre temă şi şi despre modul în care se furnizează WordPress – la începutul fişierului.



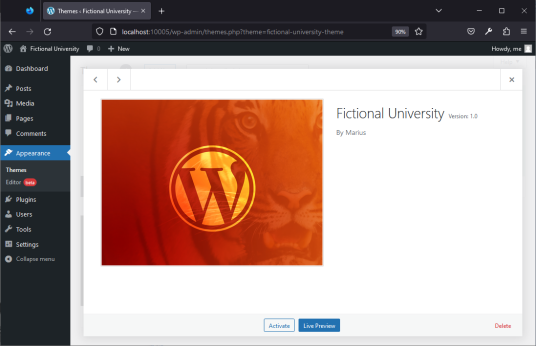
* Reîmprospătarea panoului de administrare afişează tema nou creată:

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

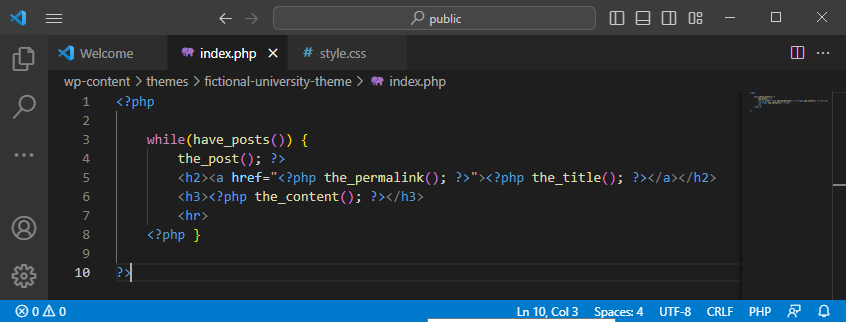
* Pentru o imagine de previzualizare a site-ului, se adăugă fişierul respectiv în folder-ul themes şi se redenumeşte „screenshot.png”.



* 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)

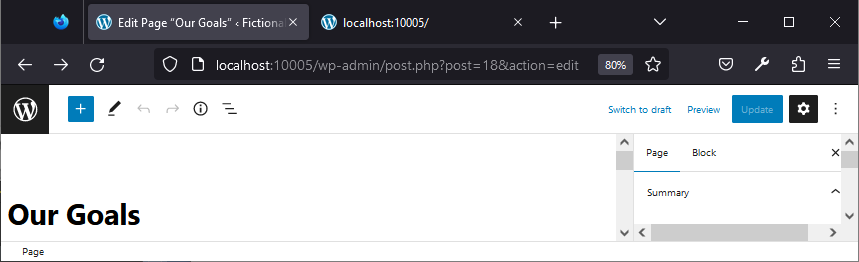
* Se creează fişierele header.php şi footer.php în partea principală a site-ului pentru antete şi subsoluri globale.
* Se folosesc funcţiile get\_header() şi get\_footer().
* Funcţia WordPress get\_head() încarcă fişierul header.php.
* Funcţia WordPress wp\_footer() încarcă bara de navigare superioară 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’); ?>”>

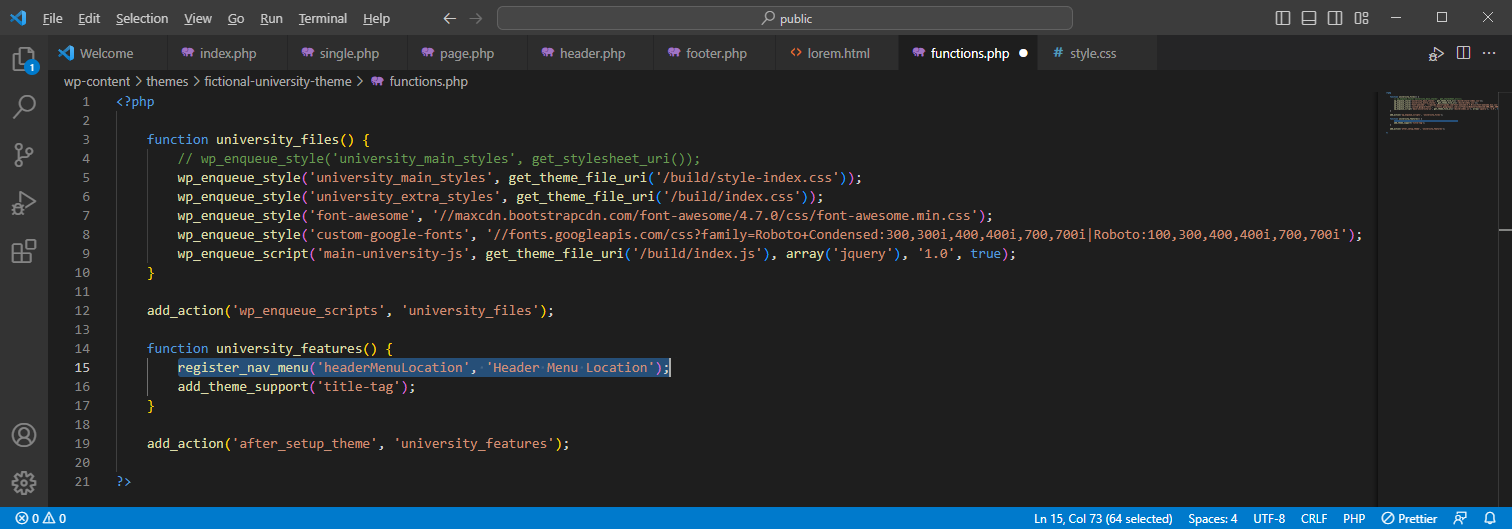
* Se generează codul HTML <meta charset=”UTF-8”>.
* Prin folosirea codului PHP în tag-ul body

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

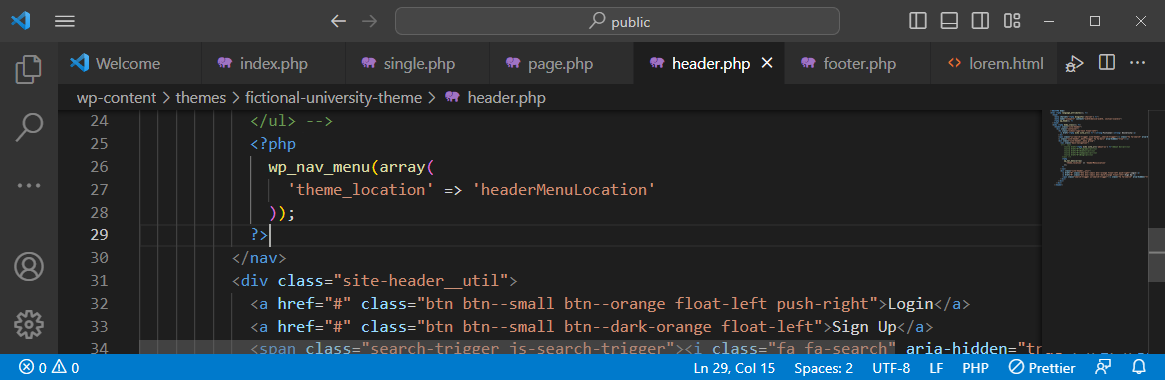
în codul HTML sunt generate mai multe clase în funcţie de contextul paginii.

## 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

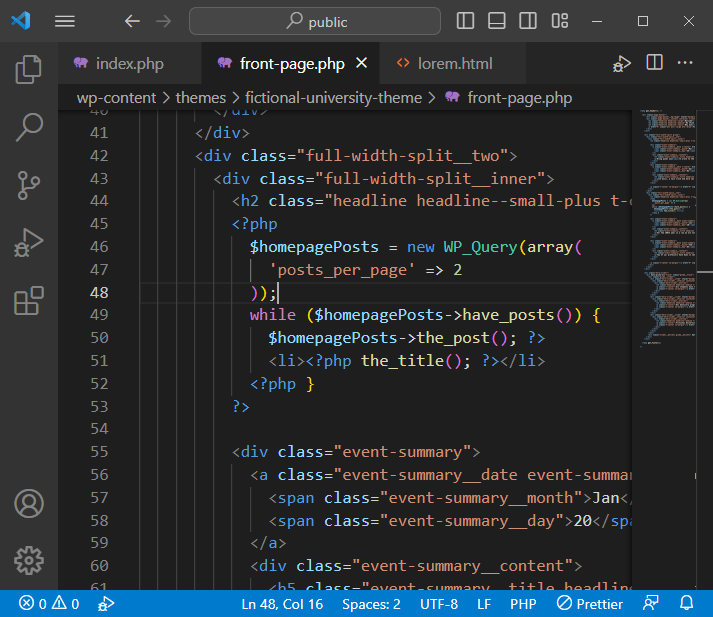
* Implicit, WordPress face pagina de home o listare generică de blog – afişează toate postările de blog mai recente.
* Dacă s-a creat o temă nouă, fără pagină de blog, se poate crea una din pagina de administrare WordPress.
* În secţiunea Settings > Reading > Front page displays se poate specifica ce 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

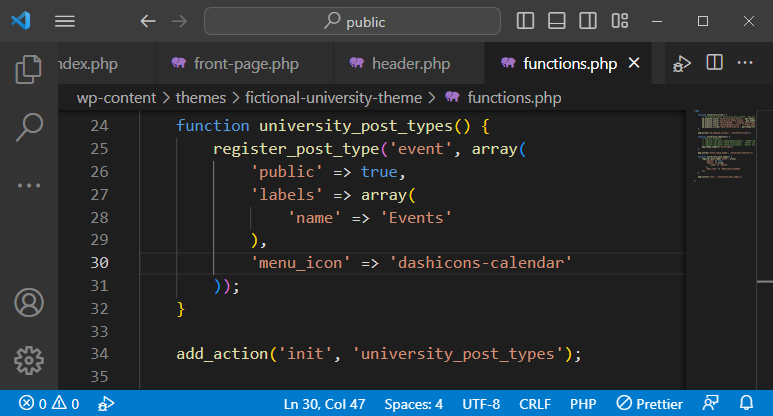
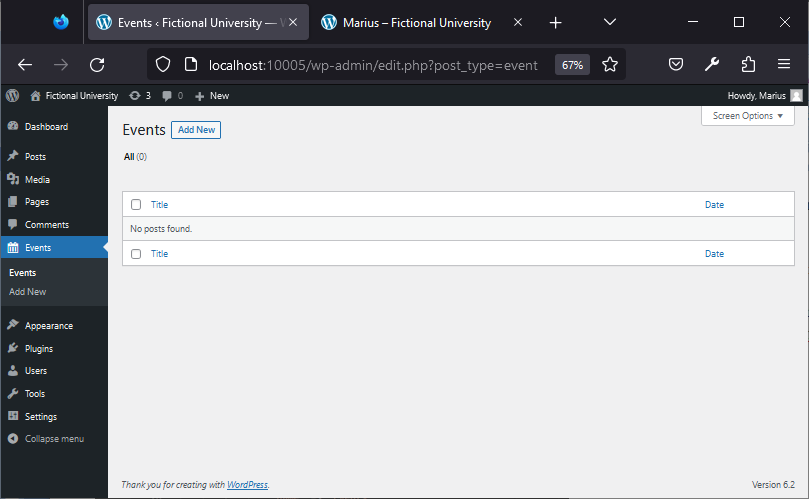
* Interogările normale (implicite) ale bazei de date se fac automat de către WordPress pe baza specificului URL-ului vizitat.
* Interogările personalizate permit încărcarea a orice, oriunde.
* Înaintea creării unei interogări personalizate, se creează o variabilă care să reprezinte numele interogării 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