# Ministerul Educaţiei şi Tineretului al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică şi Microelectronică

Catedra Automatica şi Tehnologii Informaţionale

Raport

Programarea în Windows

Lucrarea de laborator Nr. 4

Tema: Web development

A efectuat: studentul grupei TI-141: **Buldumac Vasile**

A verificat: lector asistent: **Irina Cojanu**

lector superior: **Svetlana Cojocaru**

## Web development

### Prerequisites:

* IDEs/text editors (alege unul): Sublime Text 3, RubyMine, PyCharm, Komodo, Coda, phpStorm, webStorm, Brackets
* Limbaje de programare: Ruby, Python, JS, PHP
* Technologii si Frameworks: Ruby on Rails, Django, Node.js && Express

### Obiective:

* Realizarea unui simplu Web Site personal
* Familiarizarea cu HTML si CSS
* Interactiuni Javascript

### Conditii Generale:

Se considera ca ai trecut cu succes laboratorul daca ai urmat toti pasii din:

1. [Submission Process](https://github.com/BestMujik/MIDPS-labs/blob/master/Submission Process.md)
2. Trebuie sa elaborezi un program prototip care il vei arata in timpul laboratorului
3. Ai respectat DL (data limita)

### Technical Prerequisites:

* Foloseste MVC (Model–View–Controller) pattern

### Laboratory Requirements:

* Basic Level (nota 5 || 6) :
  + Realizeaza un mini site cu 3 pagini statice
* Normal Level (nota 7 || 8):
  + Site-ul trebuie sa pastreze toata informatia intr-o baza de date
* Advanced Level (nota 9 || 10):
  + Site-ul trebuie sa contina AJAX Requests.
  + Implimentarea XHR sau JSON responses. Careva din informatie trebuie sa fie dinamic incarcata pe pagina.

Mersul lucrării

**IDE**: PHPStorm

**OS:** Ubuntu 14.04

Limbajul de programare: **PHP 5.5**

Pentru crearea rapidă a unui simplu web site personal am ales limbajul de programare PHP. Am realizat pattern-ul MVC. Model-view-controller (MVC) (din engleză, aproximativ: model-față-controlor) este un model arhitectural utilizat în ingineria software. Succesul modelului se datorează izolării logicii de business față de considerentele interfeței cu utilizatorul, rezultând o aplicație unde aspectul vizual sau/și nivelele inferioare ale regulilor de business sunt mai ușor de modificat, fără a afecta alte nivele.

Javascript: **jQuery 2.2.2**

Pentru manipularea cu structura proiectului am utilizat biblioteca jquery. Prin intermediul careia am utilizat tehnologia ajax ca sa fie indeplinita cerinta de manipulare dinamica cu datele.

Ajax (sau AJAX), prescurtare pentru Asynchronous JavaScript and XML, este o tehnică de programare pentru crearea de aplicații web interactive. Intenția este să facă paginile web să devină mai rapide și deci mai acceptate, prin schimbul în fundal al unor cantități mici de date cu serverul, astfel încât să nu fie nevoie ca pagina să fie reîncărcată la fiecare acțiune a utilizatorului. Aceasta are ca scop creșterea interactivității, vitezei și ușurinței în utilizare a aplicațiilor web.

Ajax nu este o tehnologie în sine. Termenul este folosit pentru definirea aplicațiilor web ce folosesc un ansamblu de tehnologii:

* HTML sau XHTML pentru structura semantică a informațiilor;
* CSS pentru prezentarea informațiilor;
* Javascript pentru interactivitate, pentru procesarea informațiilor prezentate;
* Obiectul XMLHttpRequest pentru schimbul și manipularea informațiilor într-o manieră asincronă cu server-ul web;
* XML este folosit de obicei pentru transferarea datelor între server și client, deși orice format funcționează, inclusiv HTML preformatat, text simplu etc.

Template Engine: **Miranda 1.0**

Este o biblioteca scrisă în PHP care ne permite utilizarea variabileleor si instrucțiunilor PHP într-o formă mai simplă și mai apropiată de către utilizator. Proiectul fiind alcătuit dîn mai multe secvente de cod care îi creaiază structura, Template Engine Miranda este folosit pentru a ușura “compilarea” acestor secvențe într-un singur fișier-structură.

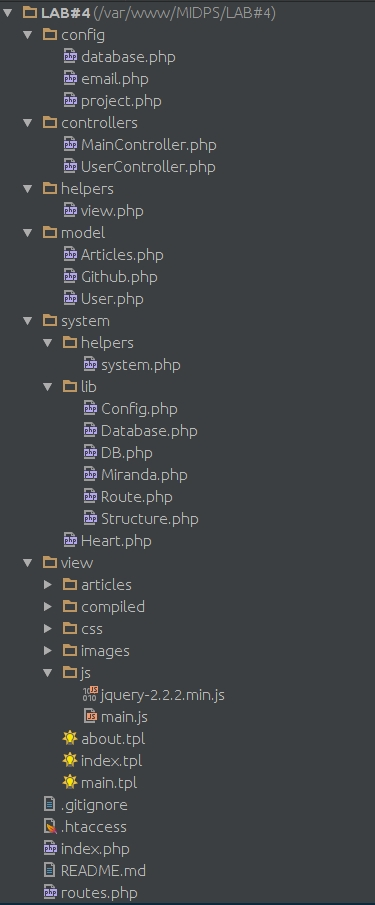
**Instalare:**

În mapa cu conținutul proiectului se îndeplinește în terminal:

*>>>* ***cd*** */adresa/directa/la/mapa/proiectului*

*>>>* ***php*** *-S 127.0.0.1:9999*  
  
Acum putem accesa proiectul prin adresa: *http://127.0.0.1:9999*

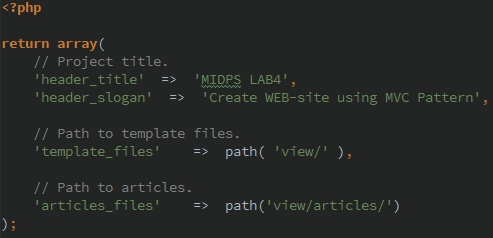
Structura proiectului



Descrierea structurii:

**config** – mapa în care sunt păstrate fișiere de tip PHP, care returneaza un array prin intermediuk clasei Config::get( ' <denumirea fisierului fara extensie> . < key-a tabloului > ' );

**config / project.php** – fișierul cu setările de bază a proiectului nostru. Aici este dată atresa către mapa cu fișierele “template” și adresa către mapa “Database” care răspunde de stocarea articolelor în php fișiere cu conținut **JSON**. Tot în acest fișier este indicat title al proiectului.



**Controllers** – Aici sunt salvate clasele în care se îndeplinește primirea datelor de la utilizator, trimiterea datelor către Model apoi afișarea datelor în View. Executarea acestor clase din mapa controllers are loc în urma apelării lor de către Route.php

controllers / MainController.php – În acest controller se îndeplinește toată logica proiectului dat. În moment ce utilizatorul acesează adresa proiectului ( 127.0.0.1 ) se apelează metoda index() din controller.

**Metoda static :: about()**

Se apelează în moment ce utilizatorul accesează adresa 127.0.0.1/about în ea are încărcarea fișierului view “about.tpl” care este salvat în mapa /view/.

**Metoda static :: new\_article()**

Se îundeplinește prin intermediul tehnologiei **AJAX**. Ei îi sunt transmise datele de tip **JSON**, care mai apoi vor fi salvate în baza de date bazată pe fișiere din mapa ( /view/articles/\*.php ).

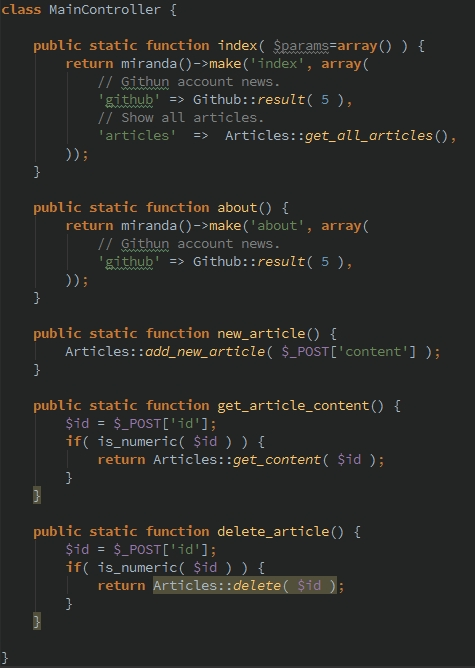
**Metoda static :: get\_article\_content()**

Se îndeplinește cu ajutorul tehnologiei **AJAX**, în moment ce se trimite cerere către server la adresa

127.0.0.1/get\_article\_content îi este transmis prin POST ID-ul articolului textul căruia avem nevoie.

**Metoda static :: delete\_article()**

Se îndeplinește prin intermediul unui AJAX-Request către adresa 127.0.0.1/delete\_article și are loc ștergerea unui fișier în dependență de ID-ul articolului.



În Controller este folosit helper-ul **miranda()** care inițializează biblioteca Miranda ce ne permite încărcarea fișierelor “view” pentru a reda structura proiectului în frontend. În View este trimis un array care salvează variabilele disponibile în fișierul-template.

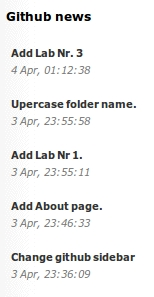
**Helpers / view.php** – funcțiile ajutătoare. În cazul dat este doar o funcție care îndeplineste inițializarea bibliotecii Miranda pentru a lucra cu fușierele “template” a proiectului.



Atributul principal în contructorul class-ei Miranda este adresa unde sunt stocate fișierele “template”. În cazul dat fișierele sunt salvate în mapa view, apoi fișierele de tip \*.tpl sunt compilate în PHP în mapa view/compiled pentru ca mai apoi să fie executate direct.

**Models / Github,php** – Această class-ă răspunde de **afișarea ultimilot N-commituri de pe contul utilizatorului github** în dependență de lincul dat de către utilizator.

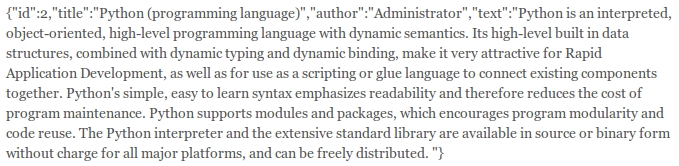
Rezultatul îl putem vedea în sidebar “Github news”.



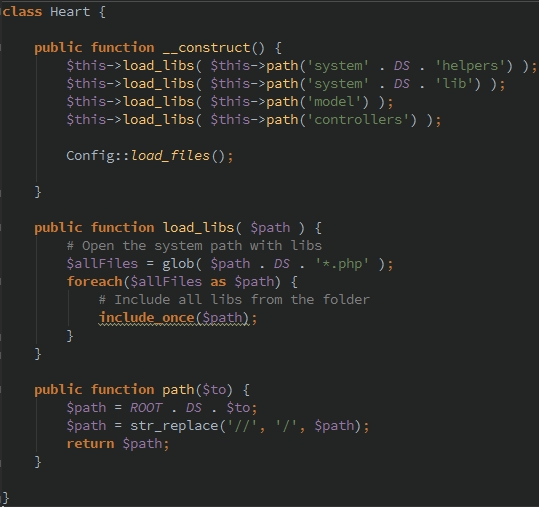
Cu ajutor funcției file\_get\_contents primim source-code la pagina utilizatorului pe github, apoi cu ajutorul la regular-expresion structurăm datele obtinute pentru a le manipula mai apoi pe pagină noastră. Salvăm doar **titile, link** și **data** îndeplinirii commit-ului.

**Models / Articles.php** – îndeplinește rolul la bază de date. Prin ea are loc salvarea datelor în fișiere de tip php. Salvarea datelor are loc în mapa view/articles cu extensia php. Datele în fișiere sunt de tip JSON. Fiecare articol are un ID unical care este indicat în denumirea sa cît și în conținutul tabloului în JSON.

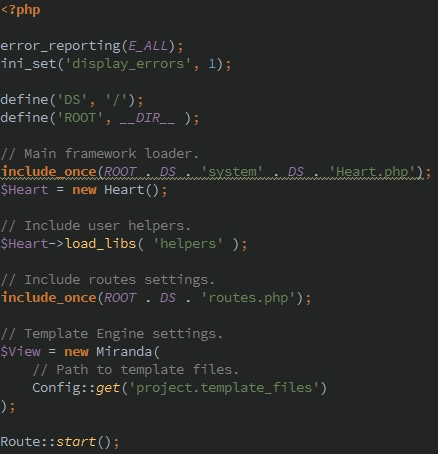
Exemplu a unui articol salvat (JSON)



**System / Heart.php** – îndeplinește funcția de bază. Prin intermediul acestui class are loc includerea bibliotecilor, și îndeplinirea pattern-ului MVC.



**Index.php** – Fișierul index.php este accesat primul, în el se îndeplinește conectarea la bibliotecile necesare și inițializarea lor.



Routes.php – Fișierul dat îndeplinește o metodă de înregistrarea a controller-lor pentru a îi apela în dependență de accesare a urilizatorului a adresei respective.

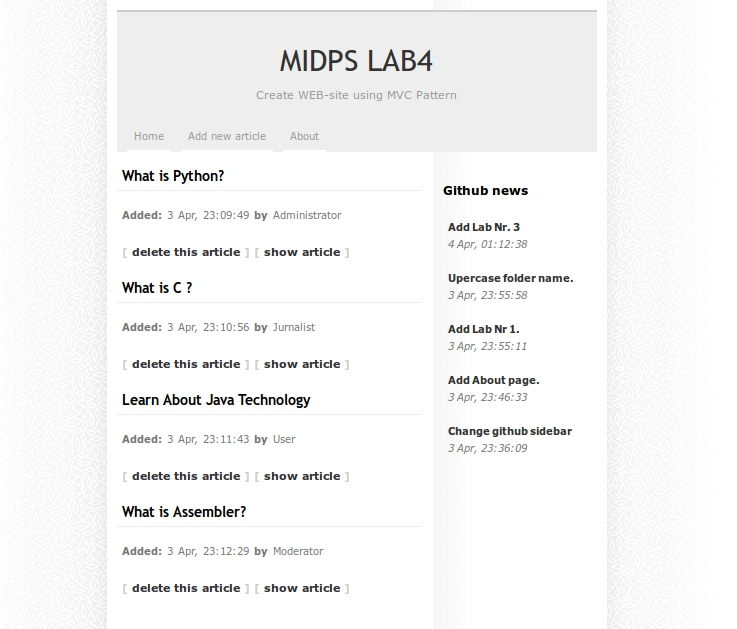


Pentru accesare adresei 127.0.0.1 **/**  - va fi apelat controller-ul MainController.

Pentru accesare adresei 127.0.0.1 **/ home** - va fi apelat controller-ul MainController.

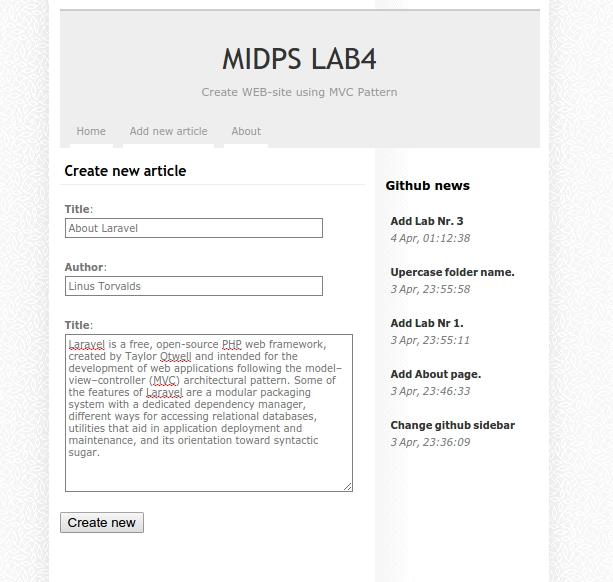
Pentru accesare adresei 127.0.0.1 **/ user** - va fi apelat controller-ul UserController.

Front-end

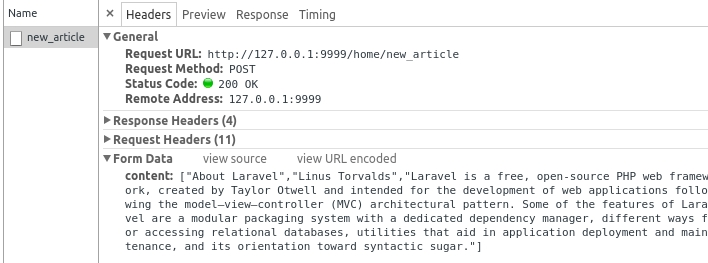


Pe pagina principală sunt afișate toate articolele create de către utilizator. În menu avem 2 link-uri statice Home și About. Și un link care se încarcă dinamic “Add new article”. În care are loc crearea unui nou articol în baza de date.

**Crearea unui nou articol** ( Ajax + Json )



Răspunsul primit în **Chrome Network**.



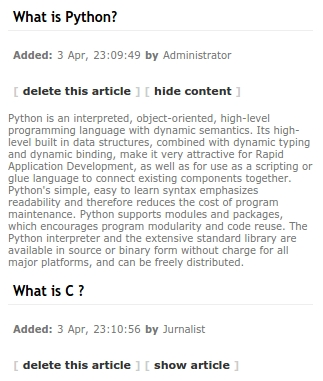
Partea jQuery (ajax) care îndeplinește salvarea articolului.



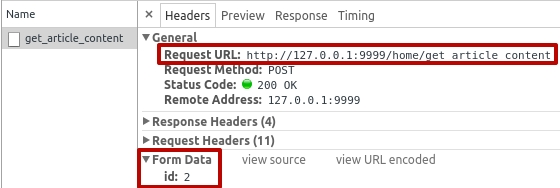
Se apelează adresa *127.0.0.1 / home / new\_article* pentru care răspunde controller-ul **HomeController** și metoda **static :: new\_article**( ) .

Încarcarea dinamică a conținutului ( jQuery + Ajax )

Cînd pagina articolele sunt afișate pe pagina principală, conținutul lor nu este încărcat în pagină. Pentru a încărca textul articolului în div-ul respectiv al articolului, trebuie să facem click pe link-ul “show article”.



Răspunsul primit în **Chrome Network**.



Partea jQuery + Ajax ( Afisare a continutului )



Se transmit adresei **127.0.0.1 / home / get\_article\_content** ID-ul articolului prin intermediul POST. Apoi rezultatul obtinut se include în DIV-ul corespunzător articolului.

Partea jQuery + Ajax ( Ștergerea articolului )



**Concluzie**

Am însușit modalitățile de creare a unui web-site cu conținut static și dinamic. Am studiat biblioteca jQuery care este un set de metode care ne permite să folosim instrucțiuni mai scurte și mai înțelese de cît le ar fi scris în Javascript curat. Am însușit posibilitățile de interactiune cu tehnologia Ajax pentru manipularea dinamică a datelor. Am creat pattern-ul MVC în PHP pentru structurarea logicei programului.

**Bibliografie**

<http://mowshon.ru/page/shablonizator-miranda-vyhod-v-svet>

<http://mowshon.ru/page/shablonizator-miranda-primery-raboty-chast-2>

<http://mowshon.ru/page/shablonizator-miranda-razdelenie-na-bloki-podkljuchenie-fajlov-chast-3>