

Ministerul Educației și Tineretului al Republicii Moldova
Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică
Catedra Automatica și Tehnologii Informaționale

Raport

MIDPS

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 Cojocar**

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
- Tehnologii 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](#)
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 bibliotecă scrisă în PHP care ne permite utilizarea variabilelor și instrucțiunilor PHP într-o formă mai simplă și mai apropiată de către utilizator. Proiectul fiind alcătuit din mai multe secvențe de cod care îi creează structura, Template Engine Miranda este folosit pentru a ușura “compilarea” acestor secvențe într-un singur fișier-structură.

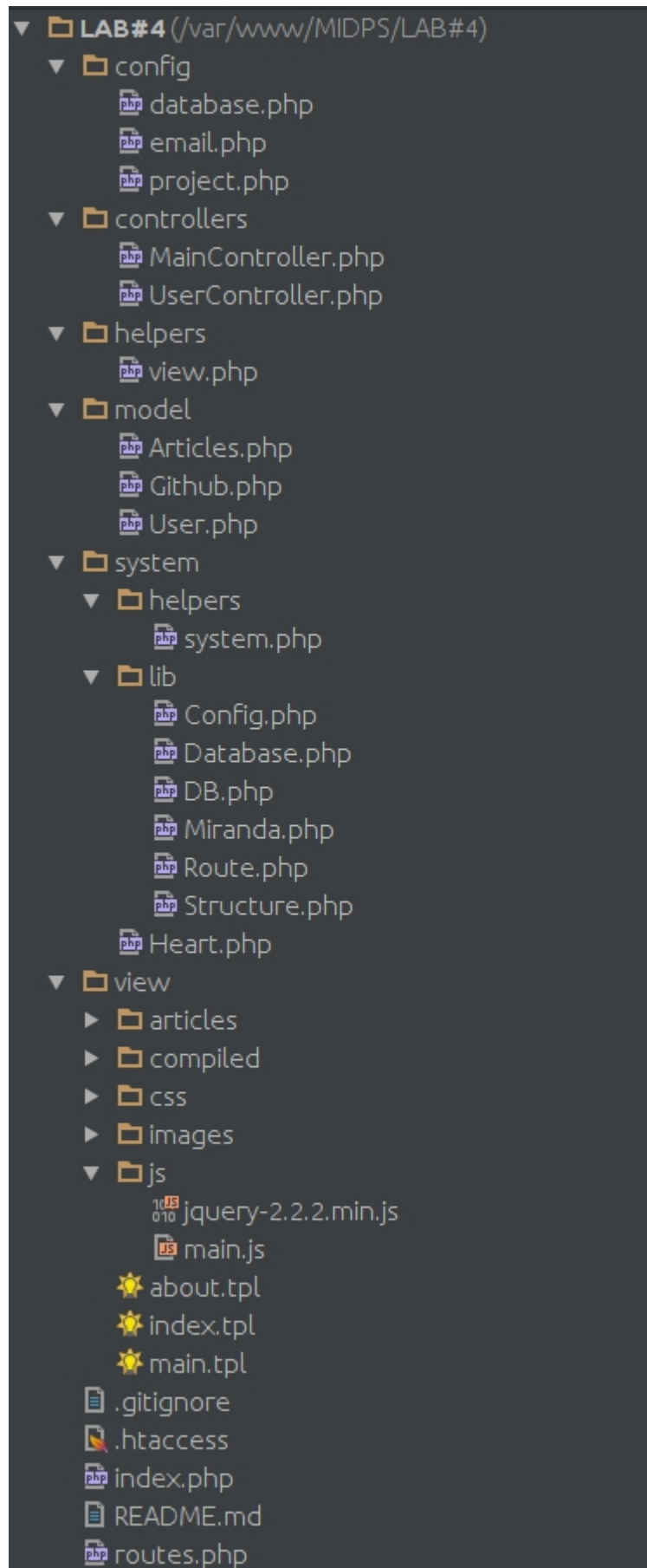
Instalare:

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 returnează un array prin intermediul clasei Config::get(' <denumirea fișierului fara extensie> . < key-a tabloului > ');

config / project.php – fișierul cu setările de bază a proiectului nostru. Aici este dată adresa 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.

```
<?php
return array(
    // Project title.
    'header_title' => 'MIDPS LAB4',
    'header_slogan' => 'Create WEB-site using MVC Pattern',

    // Path to template files.
    'template_files' => path( 'view/' ),

    // Path to articles.
    'articles_files' => path('view/articles/')
);
```

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 accesează 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 îndeplinește 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.

```

class MainController {

    public static function index( $params=array() ) {
        return miranda()->make('index', array(
            // Github account news.
            'github' => Github::result( 5 ),
            // Show all articles.
            'articles' => Articles::get_all_articles(),
        ));
    }

    public static function about() {
        return miranda()->make('about', array(
            // Github account news.
            'github' => Github::result( 5 ),
        ));
    }

    public static function new_article() {
        Articles::add_new_article( $_POST['content'] );
    }

    public static function get_article_content() {
        $id = $_POST['id'];
        if( is_numeric( $id ) ) {
            return Articles::get_content( $id );
        }
    }

    public static function delete_article() {
        $id = $_POST['id'];
        if( is_numeric( $id ) ) {
            return Articles::delete( $id );
        }
    }
}

```

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.

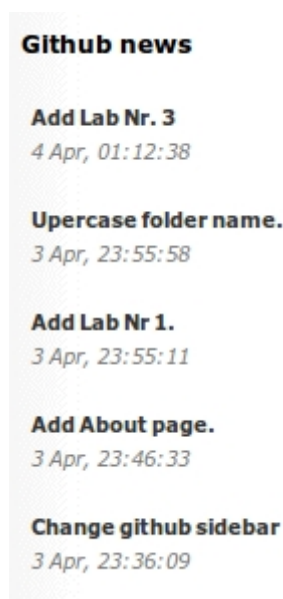
Helpers / view.php – funcțiile ajutătoare. În cazul dat este doar o funcție care îndeplinește inițializarea bibliotecii Miranda pentru a lucra cu fișierele “template” a proiectului.

```
function miranda() {  
    return new Miranda(  
        // Path to template files.  
        Config::get('project.template_files')  
    );  
};
```

Atributul principal în constructorul 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 ultimilor N-commituri de pe contul utilizatorului github** în dependență de lincul dat de către utilizator.

Rezultatul îl putem vedea în sidebar “Github news”.



Github news

- Add Lab Nr. 3**
4 Apr, 01:12:38
- Upercase folder name.**
3 Apr, 23:55:58
- Add Lab Nr 1.**
3 Apr, 23:55:11
- Add About page.**
3 Apr, 23:46:33
- Change github sidebar**
3 Apr, 23:36:09

Cu ajutor funcției `file_get_contents` primim source-code la pagina utilizatorului pe github, apoi cu ajutorul la regular-expresion structurăm datele obținute pentru a le manipula mai apoi pe pagină noastră. Salvăm doar **title**, **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 unic care este indicat în denumirea sa cât și în conținutul tabloului în JSON.

Exemplu a unui articol salvat (JSON)

```
{"id":2,"title":"Python (programming language)","author":"Administrator","text":"Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. "}
```

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

```
class Heart {

    public function __construct() {
        $this->load_libs( $this->path('system' . DS . 'helpers') );
        $this->load_libs( $this->path('system' . DS . 'lib') );
        $this->load_libs( $this->path('model') );
        $this->load_libs( $this->path('controllers') );

        Config::load_files();
    }

    public function load_libs( $path ) {
        # Open the system path with libs
        $allFiles = glob( $path . DS . '*.php' );
        foreach($allFiles as $path) {
            # Include all libs from the folder
            include_once($path);
        }
    }

    public function path($to) {
        $path = ROOT . DS . $to;
        $path = str_replace('\\\\', '/', $path);
        return $path;
    }
}
```


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

```
<?php

error_reporting(E_ALL);
ini_set('display_errors', 1);

define('DS', '/');
define('ROOT', __DIR__ );

// Main framework loader.
include_once(ROOT . DS . 'system' . DS . 'Heart.php');
$Heart = new Heart();

// Include user helpers.
$Heart->load_libs( 'helpers' );

// Include routes settings.
include_once(ROOT . DS . 'routes.php');

// Template Engine settings.
$view = new Miranda(
    // Path to template files.
    Config::get('project.template_files')
);

Route::start();
```

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

```
Route::add('/', 'MainController');
Route::add('home', 'MainController');
Route::add('user', 'UserController');
```

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)

MIDPS LAB4

Create WEB-site using MVC Pattern

Home Add new article About

Create new article

Title:

Author:

Title:

Laravel is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model-view-controller (MVC) architectural pattern. Some of the features of Laravel are a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar.

Create new

Github news

Add Lab Nr. 3
4 Apr, 01:12:38

Uppercase folder name.
3 Apr, 23:55:58

Add Lab Nr 1.
3 Apr, 23:55:11

Add About page.
3 Apr, 23:46:33

Change github sidebar
3 Apr, 23:36:09

Răspunsul primit în **Chrome Network**.

Name	×	Headers	Preview	Response	Timing
<input type="checkbox"/> new_article					
▼ General					
Request URL: http://127.0.0.1:9999/home/new_article					
Request Method: POST					
Status Code: 200 OK					
Remote Address: 127.0.0.1:9999					
► Response Headers (4)					
► Request Headers (11)					
▼ Form Data					
view source					
view URL encoded					
content: ["About Laravel","Linus Torvalds","Laravel is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model-view-controller (MVC) architectural pattern. Some of the features of Laravel are a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar."]					

Partea jQuery (ajax) care îndeplinește salvarea articolului.

```
// Add new Article.
$('.create-article').on( 'click', function() {
    var title = $('input[name=title]').val();
    var author = $('input[name=author]').val();
    var text = $('textarea').val();

    if( ! title ) { alert( 'Title is empty!' ); return; }
    if( ! author ) { alert( 'Author is empty!' ); return; }
    if( ! text ) { alert( 'Text is empty!' ); return; }

    $.ajax({
        type: 'POST',
        url: '/home/new_article',
        data: 'content=' + JSON.stringify( [title, author, text] ),
        success: function(data){
            // Success!
            alert( 'Article was created with success!' );
            $('input[type=text]').val('');
            $('textarea').val('');
        }
    });
});
```

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”.

What is Python?

Added: 3 Apr, 23:09:49 by Administrator

[delete this article] [hide content]

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

What is C ?

Added: 3 Apr, 23:10:56 by Jurnalist

[delete this article] [show article]

Răspunsul primit în **Chrome Network**.

Name	×	Headers	Preview	Response	Timing
<input type="checkbox"/> get_article_content	▼	General			
		Request URL: http://127.0.0.1:9999/home/get article content			
		Request Method: POST			
		Status Code: 200 OK			
		Remote Address: 127.0.0.1:9999			
		▶ Response Headers (4)			
		▶ Request Headers (11)			
		▼ Form Data	view source	view URL encoded	
		id: 2			

Partea jQuery + Ajax (Afisare a continutului)

```
// Show article content.
$('.article-status').on( 'click', function() {
    var article_id = $(this).data('id');

    if( 'show article' == $(this).text() ) {
        $(this).text('hide content')
        // Get article content with json.
        $.ajax({
            type: 'POST',
            url: '/home/get_article_content',
            data: 'id=' + article_id,
            success: function(data){
                $('.article-' + article_id + '-content').removeClass('hide');
                $('.article-' + article_id + '-content').text( data );
            }
        });
    } else {
        $(this).text('show article')
        $('.article-' + article_id + '-content').addClass('hide');
    }
});
```

Se transmite 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)

```
// Delete article.
$('.delete-me').on( 'click', function() {
    var article_id = $(this).data('id');
    var agree = confirm( "Do you really want to delete this article?" );
    if( agree ) {
        $.ajax({
            type: 'POST',
            url: '/home/delete_article',
            data: 'id=' + article_id,
            success: function(data){
                $('.article-' + article_id).addClass('hide');
                alert( "Article with ID:" + article_id + " is gone." );
            }
        });
    }
});
```

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 cât le ar fi scris în Javascript curat. Am însușit posibilitățile de interacțiune 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>