**Лабораторна робота №1**

**Тема: Система контролю версій GIT.**

**Мета: Навчитись працювати у git, trello та на сайті github.com.**

Хід роботи

Першим кроком було встановлення git з офіційного сайту.

http://git-scm.com/downloads

Після установки тім-лідер створив репозиторій на github.com. Відповідно наступним кроком є реєстрація на сайті, вхід у організацію даного проекту та fork загального репозиторію:

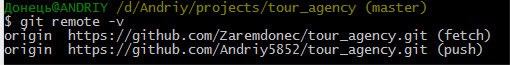


Після форку репозиторію на свій акаунт налаштуємо роботу з github.com, щоб отримувати дані з загального репозиторію а надсилати дані у власний репозиторій, виконаємо команду:

git remote set-url origin https://github.com/Anfriy5852/tour\_agency.git --push

Після цього остаточно перевіримо доступо до репозиторіїв, виконаємо воманду:

git remote -v

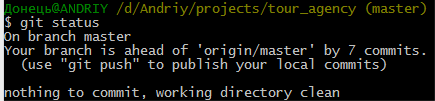


Склонуємо репозиторій:

git clone <https://github.com/Zaremdonec/tour_agency>

Наступним кроком буде саме написання коду програми, його коміти та відправлення на сервер.

Для того щоб подивитись зміни, які були внесені виконаємо команду git status.



Якщо були внесені якісь зміни, то назви файлів виведеться відповідним кольором.

Щоб зафіксувати зміни виконаємо наступні команди:

git add \* (додаємо усі файли)

git commit -m “Some commit”(коміт з назвою)

Після цього пушимо на сервер:

git push origin master (відправляємо на вітку мастер)

Після цих дій кидаємо puul request тім-ліду:



Для того щоб запушити на іншу гілку потрібно її створити

git branch “single\_item”

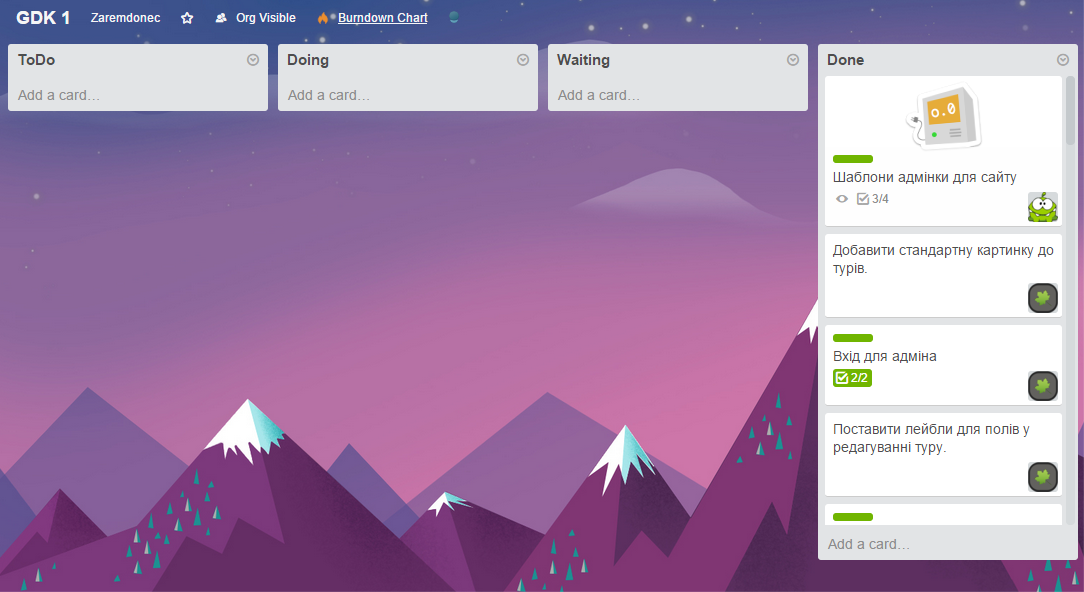
Далі перейдемо на цю гілку

git checkout single\_item

На даному рисунку зображені усі коміти проекту та гілки:



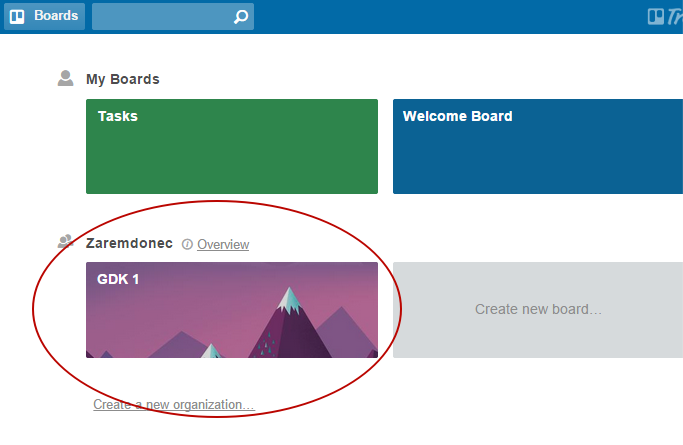
Усі завдання, та процес їх виконання контролювався на сайті Trello.com:



# Хід роботи

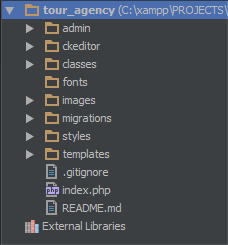
Завданням на лабораторну роботу було створити репозиторій на GitHub, координувати роботу команди та виконувати свою частину кодування.

Було створено дошку на Trello.com для зв’язку між членами команди.

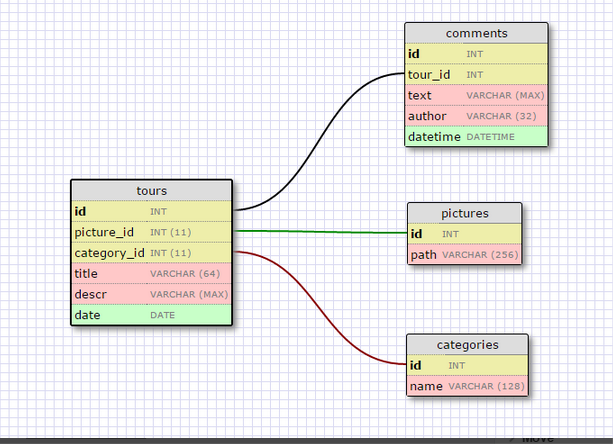


Було створене репозиторій на GitHub.com

Для зручнішої роботи з проектом було організовано його структуру:



Було змодельовано базу даних, яка має наступний вигляд:



Створено клас для роботи з базою даних, який базується на паттерні Singleton  
Код:

<?php

/\*\*

\* Using

\* $db = Database::getInstance();

\* $command = "command text";

\* $result = $db->query($command);

\*/

class Database {

protected static $\_instance;

private $\_host = "localhost";

private $\_username = "root";

private $\_password = "";

private $\_database = "gdk\_touragency";

private $\_connection;

static public function getInstance()

{

if(self::$\_instance === null) {

self::$\_instance = new self();

}

return self::$\_instance;

}

private function \_\_construct()

{

$this->\_connection = new mysqli($this->\_host, $this->\_username,

$this->\_password, $this->\_database) or die("Database connection error");

}

private function \_\_clone() { }

public function getConnection()

{

return $this->\_connection;

}

public function query($command)

{

return $this->getConnection()->query($command);

}

}

Запрограмовано перегляд окремої екскурсії, витягування даних з бази.

Код:

<?php

include('Database.php');

class Excursion

{

private $db;

private $\_title;

private $\_description;

private $\_picturePath;

private $\_city;

private $\_date;

public function \_\_construct()

{

$this->db = Database::getInstance();

}

static public function getById($id)

{

$instance = new self();

$command = "SELECT \* FROM tours WHERE tours.id=$id";

$result = Database::getInstance()->query($command);

$row = mysqli\_fetch\_array($result);

$instance->\_title = $row['title'];

$instance->\_date = $row['date'];

$instance->\_description = $row['descr'];

$instance->\_picturePath = $row['picture\_path'];

//$instance->\_city = $row['category'];

return $instance;

}

public function getTitle()

{

return $this->\_title;

}

public function getDate()

{

return $this->\_date;

}

public function getDescription()

{

return $this->\_description;

}

public function getPicturePath()

{

return $this->\_picturePath;

}

public function getCategory()

{

return $this->\_city;

}

public function createCategory($name)

{

$command = "INSERT INTO `categories`(name) VALUES ('$name')";

$this->db->query($command);

}

public function printAllCategories()

{

$categories = $this->selectCategories();

while ($row = mysqli\_fetch\_array($categories)) {

$id = $row['id'];

$name = $row['name'];

echo "<li><a href='?city=$id'>$name</a></li>";

}

}

public function printCategoryAsOptions()

{

$categories = $this->selectCategories();

while ($row = mysqli\_fetch\_array($categories)) {

$id = $row['id'];

$name = $row['name'];

echo "<option value=$id>$name</option>";

}

}

}

Клас для додавання та перегляду коментарів

<?php

/\*\*

\* Created by PhpStorm.

\* User: Vitaly

\* Date: 21.03.15

\* Time: 1:36

\*/

require\_once 'Database.php';

class Comment {

private $author;

private $text;

private $time;

public function \_\_construct($author, $text)

{

$this->author = $author;

$this->text = $text;

$dt = new DateTime();

$this->time = $dt->format("Y-m-d H:i:s");

}

public function attach($tour\_id)

{

$author = $this->author;

$text = $this->text;

$time = $this->time;

$command = "INSERT INTO comments (author, text, datetime, tour\_id)

VALUES ('$author', '$text', '$time', $tour\_id)";

Database::getInstance()->query($command);

}

static public function printAllComments($tour\_id)

{

$command = "SELECT \* FROM comments WHERE tour\_id=$tour\_id;";

$result = Database::getInstance()->query($command);

while($row = mysqli\_fetch\_array($result)) {

$author = $row['author'];

$text = $row['text'];

$time = $row['datetime'];

echo "<div class='comment'><strong>$author</strong><span>$time</span>

<div style='clear: both'></div><p>$text</p></div>";

}

}

static public function getAllComments($tour\_id)

{

$command = "SELECT \* FROM comments WHERE tour\_id=$tour\_id;";

return Database::getInstance()->query($command);

}

}

Розмітка сторінки виглядатиме наступним чином:

<div class="main\_wraper">

<?php

include("header.php");

require\_once("../classes/Excursion.php");

require\_once("../classes/Comment.php");

$tour = new Excursion();

if(isset($\_GET['id'])) {

$tour\_id = $\_GET['id'];

$tour = Excursion::getById($tour\_id);

if(isset($\_POST['post-btn'])) {

$comment = new Comment($\_POST['author'], $\_POST['text']);

$comment->attach($tour\_id);

}

}

?>

<div class="wrap">

<div class="tour\_item">

<h1>

<?= $tour->getTitle() ?>

</h1>

<div class="date">

<?= $tour->getDate() ?>

</div>

<p>

<?= $tour->getDescription() ?>

</p>

<img src="<?= $tour->getPicturePath() ?>" width="100%">

</div>

<div class="comments">

<h2>Коментарі</h2>

<?php

if(isset($\_GET['id'])) {

Comment::printAllComments($\_GET['id']);

}

?>

<hr>

<div class="leave-comment">

<form method="post" action="<?php if(isset($\_GET['id'])) echo $\_SERVER['PHP\_SELF']."?id={$\_GET['id']}"?>">

<label for="input1">Ім'я:</label>

<input id='input1' name="author" class="input">

<label for="text1">Коментар:</label>

<textarea id="text1" name="text" class="input"></textarea>

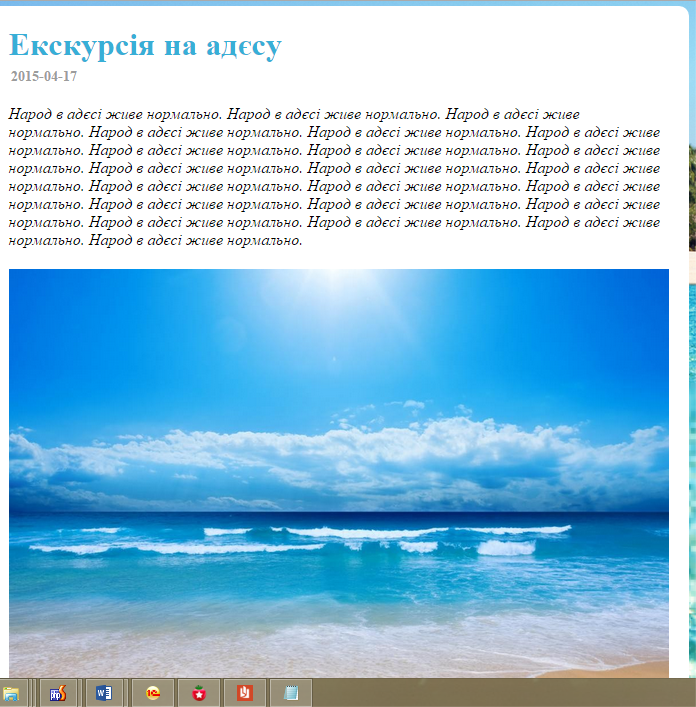
<input class="btn" type="submit" name="post-btn" value="Опублікувати">

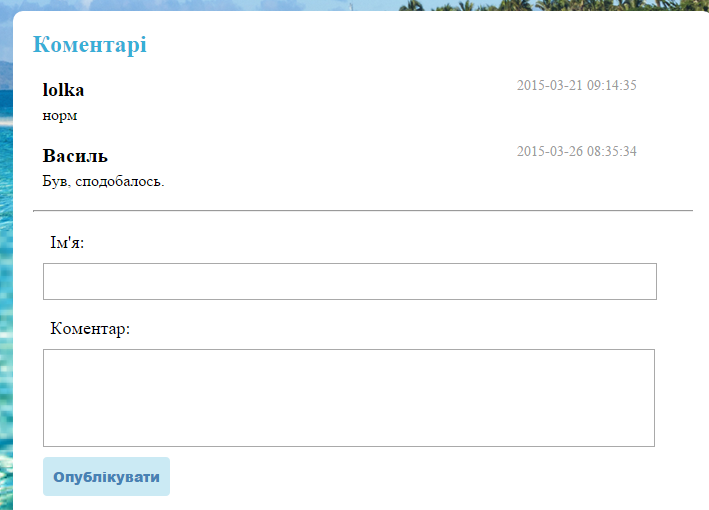
</form>

</div>

</div>

</div>





Для тестування проекту використовувався PHP фреймворк для тестування Codeception, який може бути використаний з мінімумом витрат на встановлення. Є можливість виконувати один з трьох видів тестів: Acceptance test, Functional test, Unit test. Для встановлення цього фреймворку необхідно перейти за посиланням <http://codeception.com/quickstart> та слідувати інструкціям.

За допомогою консолі установка та налаштування може виглядати наступним чином:

wget http://codeception.com/codecept.phar

php codecept.phar bootstrap

Генерація acceptance тесту з назвою WelcomeCept виглядає так:

php codecept.phar generate:cept acceptance Welcome

Для написання тесту, необхідно редагувати файл **tests/acceptance/WelcomeCept.php**

Його зміст може виглядати наступним чином:

<?php

$I = **new** AcceptanceTester($scenario);

$I->wantTo('ensure that frontpage works');

$I->amOnPage('/');

$I->see('Home');

?>

Обов’язковим в налаштуванні тесту є вказати хост сайту у файлі **tests/acceptance.suite.yml**

class\_name: AcceptanceGuy

modules:

enabled: [PhpBrowser, AcceptanceHelper]

config:

PhpBrowser:

url: '**{YOUR APP'S URL}**'

Для виконання усіх тестів необхідно виконати команду

php codecept.phar run

Та переглянути результати тестів у консолі

Також було проведено валідацію html5 документів на онлайн сервісі <http://validator.w3.org/check>

# 

Про знайдені помилки, які відносяться до html5 було повідомлено відповідальним за це особам.

# Висновки:

Під час виконання лабораторної роботи, я навчився працювати з системою контролю версій git, використовувати для створння та споглядання результатів сайт github.com, та користуватоись board-ми на сайті trello.com.