Додаток А

Лістинг програми

```
<?php
class UserTest extends VDbTestCase {
    public $fixtures=array(
        'addresses'=>'Address',
        'configs'=>'Config',
        'users'=>'User',
        'orders'=>'Order',
        'user_oauth'=>'OauthUser',
        'notifications'=>'Notification',
        'audiences'=>'Audience',
        'notification_views'=>'notificationView',
        'ads_history' => 'AdHistory',
        'questions'=>'Question',
        'answers'=>'Answer',
        'addresses'=>'Address',
        'trending_videos_votes'=>'TrendingVideoVote',
        'user_follows'=>'UserFollow',
        'user_alerts'=>'UserAlert'     );
    protected function setUp() {
        parent::setUp();
        $this->obj = new User();    }
    public function testBeforeSave() {
        $_password = '12345';
        Yii::app()->request->cookies['inviter_hash'] = new CHttpCookie('inviter_hash', 'ijkl');
       $this->obj->email = 'testuser@vidfall.com';
        $this->obj->password = $_password;
        $this->obj->save();
        $this->assertEquals($this->obj->password, mUser::encryptPassword($_password));
        $this->assertStringStartsWith(date('Y-m-d'), $this->obj->reg_date);
        $this->assertEquals(3, $this->obj->inviter_id);
        $this->assertNotEmpty($this->obj->ref_name);
        $user = User::model()->findByPk(1);
        $user->password = $_password;
        $user->auth_token = '';
```

```
    $user->save();
    $this->assertEquals($user->password, mUser::encryptPassword($_password));
    $this->assertTrue($user->auth_token != '');    }
public function testAfterSave() {
    Yii::app()->request->cookies['inviter_hash'] = new CHttpCookie('inviter_hash', 'ijkl');
    $this->obj->email = 'testuser@vidfall.com';
    $this->obj->password = '12345';
    $this->obj->save();
    $this->assertFalse(isset(Yii::app()->request->cookies['inviter_hash']));    }
public function testGetAddress() {
    $user = User::model()->findByPk(1);
    $addressBilling = $user->getAddress(Address::TYPE_BILLING);
    $this->assertEquals($addressBilling['work_phone'], '0986473828');
    $addressShipping = $user->getAddress(Address::TYPE_SHIPPING);
    $this->assertEquals($addressShipping['address2'], 'Novyi Svit 22');
    $user = User::model()->findByPk(7);
    $this->assertNull($user->getAddress(Address::TYPE_SHIPPING));    }
    public function testSetExpressCheckoutSeenDate() {
        //Notification Seen is empty
    $user = User::model()->findByPk(1);
    $user->setExpressCheckoutSeenDate();
    $this->assertEquals($user->notification_seen, date('Y-m-d'));
    //Notification Seen is not empty
    $user = User::model()->findByPk(2);
    $user->setExpressCheckoutSeenDate();
    $this->assertEquals($user->notification_seen, date('Y-m-d'));    }
public function testExpressCheckoutIsAvailable() {
    //only billing address
    $user = User::model()->findByPk(11);
    $this->assertFalse($user->expressCheckoutIsAvailable());
    //filled billing, shipping address and strip_id
    $user = User::model()->findByPk(1);
    $this->assertTrue($user->expressCheckoutIsAvailable());    }
public function testPaymentMethodIsSet() {
    //stripe_id is empty
    $user = User::model()->findByPk(7);
```

```
        $this->assertFalse($user->paymentMethodiIsSet());
        //stripe_id is not empty
        $user = User::model()->findByPk(1);
        $this->assertTrue($user->paymentMethodiIsSet());    }
    public function testCreateStripeAccount() {
        $user = User::model()->findByPk(1);
        $this->assertTrue($user->createStripeAccount('TEST'));
        $this->assertEquals('4321', $user->stripe_id);
        $this->assertEquals('5454', $user->last4);
        $this->assertEquals(Yii::t('m', 'Your card was declined'), $user->createStripeAccount('TEST
ERROR'));
        $this->assertEquals(Yii::t('m', 'bad request'), $user->createStripeAccount('TEST ERROR
REQUEST'));    }
    public function testGetInvitedUsersInfo() {
        $this->obj->id = 2;
        $this->assertSame($this->obj->getInvitedUsersInfo(), array());
        $this->obj->id = 3;
        $invitedUsers = $this->obj->getInvitedUsersInfo();
        $this->assertEquals(count($invitedUsers), 3);
        $this->assertSame($invitedUsers[0]['email'], 'some@user.com');
        $this->assertSame($invitedUsers[1]['email'], 'valjes@wasabiventures.com');
        $this->assertSame($invitedUsers[2]['email'], 'vova@wasabiventures.com');    }
    public function testGetNumberOfPurchases() {
        $this->obj->id = 1;
        $this->assertEquals($this->obj->getNumberOfPurchases(), 1);
        $this->obj->id = 2;
        $this->assertEquals($this->obj->getNumberOfPurchases(), 2);
        $this->obj->id = 3;
        $this->assertEquals($this->obj->getNumberOfPurchases(), 0);
        $this->obj->id = 5;
        $this->assertEquals($this->obj->getNumberOfPurchases(), 0);    }
    public function testFindByEmail() {
        $user = $this->obj->findByEmail('joel@vidfall.com');
        $this->assertTrue(is_object($user));
        $this->assertNull($this->obj->findByEmail('wronguser@vidfall.com'));    }
    public function testValidatePassword() {
```

```
    $user = User::model()->findByPk(2);
    $this->assertFalse($user->validatePassword('12345'));
    $this->assertTrue($user->validatePassword('111111'));    }
public function testCreatePassword() {
    $this->obj->createPassword();
    $this->assertGreaterThan(0, strlen($this->obj->password));
    $user = User::model()->findByPk(2);
    $user->createPassword();
    $this->assertTrue($user->validatePassword('111111'));    }
public function testExpressIsSet() {
    $this->obj->expressIsSet();
    $this->assertTrue($this->obj->express_was_set);    }
public function testIsBeginner() {
    //User has orders
    $user = User::model()->findByPk(1);
    $this->assertFalse($user->isBeginner());
    //User has no orders
    $user = User::model()->findByPk(3);
    $this->assertTrue($user->isBeginner());
    //User has only new orders
    $user = User::model()->findByPk(5);
    $this->assertTrue($user->isBeginner());
    //User has only canceled orders
    $user = User::model()->findByPk(12);
    $this->assertFalse($user->isBeginner());    }
public function testIsVip() {
    //Invited 10 users who created accounts
    $user = User::model()->findByPk(1);
    $this->assertTrue($user->isVip());
    $user = User::model()->findByPk(2);
    $this->assertFalse($user->isVip());
    //Invited 3 users who made a purchases
    $user = User::model()->findByPk(3);
    $this->assertTrue($user->isVip());    }
public function testCouponSent() {
    $this->obj->couponSent();
```

```
    $this->assertEquals($this->obj->coupon_sent, 1);    }
public function testHasFBlogin() {
    $this->assertTrue(User::model()->findByPk(1)->hasFBLogin());
    $this->assertFalse(User::model()->findByPk(2)->hasFBLogin());    }
public function testGetName() {
    $user = User::model()->findByPk(1);
    $name = $user->getName();
    $this->assertInternalType('array', $name);
    $this->assertEquals('John', $name['fname']);
    $this->assertEquals('Doe', $name['lname']);
    $user = User::model()->findByPk(7);
    $name = $user->getName();
    $this->assertInternalType('array', $name);
    $this->assertEquals('invited3@user.com', $name['fname']);
    $this->assertEquals('', $name['lname']);
    $user = User::model()->findByPk(1);
    $address = Address::model()->findByPk(1);
    $address->id=20;
    $address->save();
    $name = $user->getName();
    $this->assertInternalType('array', $name);
    $this->assertEquals('John', $name['fname']);
    $this->assertEquals('Doe', $name['lname']);
    $address->delete();
    $name = $user->getName();
    $this->assertInternalType('array', $name);
    $this->assertEquals('Vasya', $name['fname']);
    $this->assertEquals('Pypkin', $name['lname']);    }
public function testGetNameText() {
    $user = User::model()->findByPk(1);
    $this->assertEquals('John D', $user->getNameText());
    $user = User::model()->findByPk(7);
    $this->assertEquals('invited3@user.com', $user->getNameText());    }
public function testHOAUTHafterRegister() {
    $user = User::model()->findByPk(1);
    $rewards = $user->rewards;
```

```
        $user->hOAUTHafterRegister();
        $user->refresh();
        $this->assertEquals($user->rewards, $rewards +
mConfig::getValue(mConfig::POINTS_PER_REGISTRATION));    }
    public function testAddRewards() {
        $user = User::model()->findByPk(1);
        $user->addRewards(100);
        $user->refresh();
        $this->assertEquals(100100, $user['rewards']);
        $this->assertEquals(5100, $user['rewards_today']);
        $this->assertEquals(100100, $user['rewards_all_time']);    }
    public function testDeductRewards() {
        $user = User::model()->findByPk(1);
        $this->assertTrue($user->deductRewards(200));
        $user->refresh();
        $this->assertEquals(99800, $user['rewards']);
        $this->assertEquals(4800, $user['rewards_today']);
        $this->assertEquals(100000, $user['rewards_all_time']);
        $user = User::model()->findByPk(3);
        $this->assertFalse($user->deductRewards(100001));
        $user->refresh();
        $this->assertEquals(100000, $user['rewards']);
        $this->assertEquals(150000, $user['rewards_all_time']);
        $this->assertTrue($user->deductRewards(100000));
        $user->refresh();
        $this->assertEquals(0, $user['rewards']);
        $this->assertEquals(-100000, $user['rewards_today']);
        $this->assertEquals(150000, $user['rewards_all_time']);    }
    public function testGetRewards() {
        $user = User::model()->findByPk(1);
        $user->rewards = 111;
        $user->save();
        $this->assertEquals($user->getRewards(), 111);
        $user->rewards = 111.222;
        $user->save();
        $this->assertEquals($user->getRewards(), 111.222);    }
```

```
public function testGetRewardsToDisplay() {
    $user = User::model()->findByPk(1);
    $user->rewards = 111;
    $user->save();
    $this->assertEquals($user->getRewardsToDisplay(), 111);
    $user->rewards = 111.222;
    $user->save();
    $this->assertEquals($user->getRewardsToDisplay(), 111);    }
public function testGetRewardsAllTime() {
    $user = User::model()->findByPk(1);
    $user->rewards_all_time = 111;
    $user->save();
    $this->assertEquals($user->getRewardsAllTime(), 111);
    $user->rewards_all_time = 111.222;
    $user->save();
    $this->assertEquals($user->getRewardsAllTime(), 111.222);    }
public function testGetRewardsAllTimeToDisplay() {
    $user = User::model()->findByPk(1);
    $user->rewards_all_time = 111;
    $user->save();
    $this->assertEquals($user->getRewardsAllTimeToDisplay(), 111);
    $user->rewards_all_time = 111.222;
    $user->save();
    $this->assertEquals($user->getRewardsAllTimeToDisplay(), 111);    }
public function testGetDailyBonusDays() {
    $this->obj->bonus_duration = 525;
    $this->assertEquals(526, $this->obj->getDailyBonusDays());    }
public function testTweetedToday() {
    $user = User::model()->find();
    $this->assertFalse((bool)$user->tweeted_today);
    $user->tweetedToday();
    $this->assertTrue((bool)$user->tweeted_today);   }
public function testDailyBonusStepCompleted() {
    $this->obj = User::model()->findByPk(4);
    $this->obj->bonus_today = false;
    $this->obj->tweeted_today = 0;
```

```
    $this->obj->email_confirmed = 0;
    $this->obj->free_spins = 0;
    $this->obj->rewards = 0;
    $this->obj->dailyBonusStepCompleted();
    $this->assertEquals($this->obj->rewards, 0);
    $this->assertEquals($this->obj->free_spins, 0);
    mTrendingVideoVote::saveVote(1, 4);
    $this->obj->dailyBonusStepCompleted();
    $this->assertEquals($this->obj->rewards, 0);
    $this->assertEquals($this->obj->free_spins, 0);
    $this->obj->tweeted_today = 1;
    $this->obj->dailyBonusStepCompleted();
    $this->assertEquals($this->obj->rewards, 0);
    $this->assertEquals($this->obj->free_spins, 0);
    $this->obj->email_confirmed = 1;
    $this->obj->dailyBonusStepCompleted();
    $this->assertEquals($this->obj->rewards, 1000);
    $this->assertEquals($this->obj->free_spins, 1);
    $this->obj->bonus_today = false;
    $this->obj->dailyBonusStepCompleted();
    $this->assertEquals($this->obj->rewards, 2000);
    $this->assertEquals($this->obj->free_spins, 2);
    $this->obj->bonus_today = false;
    $this->obj->dailyBonusStepCompleted();
    $this->assertEquals($this->obj->rewards, 3000);
    $this->assertEquals($this->obj->free_spins, 3);
    for($i = 0; $i < 40; $i++) {
       $this->obj->bonus_today = false;
       $this->obj->dailyBonusStepCompleted();      }
    $this->assertEquals($this->obj->rewards, 43000);    }
public function testDailyBonusStepCompletedForNonTweetedDay() {
    $this->obj = User::model()->findByPk(1);
    $this->obj->bonus_today = false;
    $this->obj->tweeted_today = 0;
    $this->obj->email_confirmed = 1;
    $this->obj->free_spins = 0;
```

```
    $this->obj->rewards = 0;
    $obj = new mEngagement();
    $todayTweet = $obj->getTodayTweet();
    $todayTweet->date_start=date('Y-m-d', strtotime('+1 days'));
    $todayTweet->date_end=date('Y-m-d', strtotime('+2 days'));
    $todayTweet->save();
    $this->obj->dailyBonusStepCompleted();
    $this->assertEquals($this->obj->rewards, 1000);
    $this->assertEquals($this->obj->free_spins, 1);
    $this->obj->bonus_today = false;
    $this->obj->dailyBonusStepCompleted();
    $this->assertEquals($this->obj->rewards, 2000);
    $this->assertEquals($this->obj->free_spins, 2);    }
public function testIsDailyBonusCompleted() {
    $this->assertFalse($this->obj->isDailyBonusCompleted());
    $this->obj->bonus_today = true;
    $this->obj->save();
    $this->assertTrue($this->obj->isDailyBonusCompleted());    }
public function testEnableSurveys() {
    $this->obj->show_survey = false;
    $this->obj->enableSurveys(true);
    $this->assertTrue($this->obj->show_survey);
    $this->obj->enableSurveys(false);
    $this->assertFalse($this->obj->show_survey);    }
public function testConfirmHash() {
    $this->assertTrue(empty($this->obj->email_confirm_hash));
    $this->obj->email = $this->generateEmailAddress();
    $this->obj->password = $this->generateString(8);
    $hash = $this->obj->generateConfirmHash();
    $alreadyExists = User::model()->findAllByAttributes(array('email_confirm_hash' => $hash));
    $this->assertEquals(count($alreadyExists),0);
    $this->assertEquals(strlen($hash),30);
    $this->obj->setConfirmHash();
    $this->obj->refresh();
    $this->assertEquals(strlen($this->obj->email_confirm_hash),30);
```

```
        $alreadyExists = User::model()->findAllByAttributes(array('email_confirm_hash' => $this->obj->email_confirm_hash));
        $this->assertEquals(count($alreadyExists),1);    }
    public function testGetSocialMediaData(){
        $user = User::model()->findByPk(1);
        $res = $user->getSocialMediaData();
        $this->assertTrue(isset($res[User::FB_PROVIDER]));
        $this->assertEquals($res[User::FB_PROVIDER], 1);
        $this->assertEquals($res[User::TW_PROVIDER], 1);
        $user = User::model()->findByPk(2);
        $res = $user->getSocialMediaData();
        $this->assertTrue(!isset($res[User::FB_PROVIDER]));
        $this->assertEquals($res[User::TW_PROVIDER], 1);    }
    public function testaAddAdBlock() {
        $user = User::model()->findByPk(1);
        $userAdBlockCount=$user->used_adblock;
        $this->assertEquals($userAdBlockCount, 1);
        $user->addAdBlock();
        $this->assertEquals($user->used_adblock, 1 );
        $user2 = User::model()->findByPk(1);
        $user2->addAdBlock();
        $this->assertEquals($user2->used_adblock, 1 ); }
    public function testAddPayPalCount()  {
        $user = User::model()->findByPk(1);
        $userPayPalCount=$user->clicked_pay_pal;
        $this->assertEquals($userPayPalCount, 1);
        $user->addPayPalOne();
        $this->assertEquals($user->clicked_pay_pal, 2 );
        $user->addPayPalOne();
        $this->assertEquals($user->clicked_pay_pal, 1 );
        $user->addPayPalOne();
        $this->assertEquals($user->clicked_pay_pal, 3 );
        $user2 = User::model()->findByPk(1);
        $user2->addPayPalOne();
        $this->assertEquals($user2->clicked_pay_pal, 1 );    }
    public function testGetTimeAfterRegistration(){
```

```
    $user = User::model()->findByPk(1);
    $user->reg_date = date("Y-m-d",strtotime($user->reg_date)-10);
    $this->assertTrue($user->getTimeAfterRegistration()>0);
    $user->reg_date = ((int)date("Y")+1).'-10-10';
    $user->getTimeAfterRegistration();
    $this->assertFalse($user->getTimeAfterRegistration());    }
 public function testSetNickname(){
    $user = User::model()->findByPk(1);
    $user->setNickname();
    $user->refresh();
    $this->assertEquals('John D', $user->nickname);
    $user->nickname = 'Nick W';
    $user->save();
    $user->setNickname();
    $user->refresh();
    $this->assertEquals('Nick W', $user->nickname);
    $user = User::model()->findByPk(7);
    $user->setNickname();
    $user->refresh();
    $this->assertEquals('invited3', $user->nickname);    }
 public function testSetCookie(){
    $user = User::model()->findByPk(1);
    $user->setCookie();
    $user->refresh();
    $this->assertTrue(isset(Yii::app()->request->cookies['vfuid']));
    $this->assertEquals($user->auth_token, Yii::app()->request->cookies['vfuid']->value);    }

public function testGetProfilePercentage() {
    $user = User::model()->findByPk(11);
    $this->assertEquals(0, $user->getProfilePercentage());
    $this->assertEquals(0, $user->getProfilePercentage(false));
    $user->image = 'http://b.com/a.jpg';
    $this->assertEquals(15, $user->getProfilePercentage());
    $this->assertEquals(0, $user->getProfilePercentage(false));
    $user->fname = 'fname';
    $this->assertEquals(20, $user->getProfilePercentage());
```

```
$this->assertEquals(0, $user->getProfilePercentage(false));
$user->lname = 'lname';
$this->assertEquals(25, $user->getProfilePercentage());
$this->assertEquals(25, $user->getProfilePercentage(false));
$user->nickname = 'nickname';
$this->assertEquals(30, $user->getProfilePercentage());
$this->assertEquals(25, $user->getProfilePercentage(false));
$user->quote = 'favorite_quote';
$this->assertEquals(35, $user->getProfilePercentage());
$this->assertEquals(25, $user->getProfilePercentage(false));
$settingsInterest = new SettingsInterest();
$settingsInterest->user_id = 11;
$settingsInterest->settings_type_id = 1;
$settingsInterest->save();
$this->assertEquals(50, $user->getProfilePercentage());
$this->assertEquals(50, $user->getProfilePercentage(false));
//billing address is already in fixtures, let's add shippingbg one
$address = new Address();
$address['fname'] = 'fname';
$address['lname'] = 'lname';
$address['address'] = 'address 25';
$address['city'] = 'city';
$address['country'] = 'USA';
$address['state'] = 'Texas';
$address['zip'] = '46000';
$address['home_phone'] = '46000234234';
$address['type'] = Address::TYPE_SHIPPING;
$address['user_id'] = 11;
$address->save();
$user->last4 = '0000';
$user->stripe_id = 'cus_3123424234';
$this->assertEquals(75, $user->getProfilePercentage());
$this->assertEquals(75, $user->getProfilePercentage(false));
$user->email_confirmed = true;
$this->assertEquals(100, $user->getProfilePercentage());
$this->assertEquals(100, $user->getProfilePercentage(false));    }
```

```
package com.wv.vf.ui;
import java.util.Random;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.htmlunit.HtmlUnitDriver;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.support.ui.Select;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.interactions.Action;
import org.junit.*;
import org.apache.commons.io.FileUtils;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Date;
import java.text.*;
import java.io.*;
import java.util.Properties;
import java.util.regex.Pattern;
import java.util.concurrent.TimeUnit;
import static org.openqa.selenium.OutputType.*;
public class testLoginLogout {
   private static String nickname = randomString();
   private static String email = nickname + "@gmail.com";
   private static String password = randomString();
   public static WebDriver driver = new FirefoxDriver();
   @Before
   public void setUp()    {
      driver.manage().deleteAllCookies();
      driver.manage().timeouts().implicitlyWait(25, TimeUnit.SECONDS);    }
   @Test
   public void testLoginAndLogout()    {
      Properties prop = new Properties();
```

```
        InputStream input = null;
        try       {
            String OS = System.getProperty("os.name").toLowerCase();
            if(OS.indexOf("win") >= 0) {
                input = new FileInputStream("config.properties");
            } else{
                input = new FileInputStream("config-jenkins.properties");          }
            prop.load(input);
        } catch (IOException ioe)     {
            ioe.printStackTrace();       }
        driver.navigate().to(prop.getProperty("url"));
        register();
        logOut();
        login();    }
    @After
    public void tearDown()   {
        driver.quit();    }
    public static void register()    {
        driver.findElement(By.cssSelector("a.sign_up")).click();
        driver.findElement(By.id("email")).sendKeys(email);
        driver.findElement(By.id("password")).sendKeys(password);
        driver.findElement(By.id("password_again")).sendKeys(password);
        driver.findElement(By.cssSelector("a.btn-private-register")).click();
        Actions builder = new Actions(driver);
        WebElement element = driver.findElement(By.xpath("//li[contains(@id, 'menu-profile')]"));
        driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
        WebElement subElement = driver.findElement(By.xpath("//a[contains(@href, '/user/user-
profile/')]"));
        driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
        builder.moveToElement(element).perform();
        try       {
            Thread.sleep(2000);
        } catch (InterruptedException ie)       {
            System.out.println("Thread dont work");       }
        builder.click(subElement).perform();
              try       {
```

```
            Assert.assertTrue(driver.findElement(By.xpath("//input[contains(@value, '" + nickname + "')]")).isDisplayed());
        } catch (NoSuchElementException nsee)       {
            Assert.assertFalse(true);       }    }
    public static void login()    {
        driver.findElement(By.xpath("//div[@class='header-btns-v5']/a")).click();
        driver.findElement(By.id("username")).sendKeys(email);
        driver.findElement(By.xpath("(//input[@id='password'])[2]")).sendKeys(password);
        driver.findElement(By.cssSelector("a.btn-private-login")).click();
        driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
        Actions builder = new Actions(driver);
        WebElement element = driver.findElement(By.xpath("//li[contains(@id, 'menu-profile')]"));
        WebElement subElement = driver.findElement(By.xpath("//a[contains(@href, '/user/user-profile/')]"));
        builder.moveToElement(element).perform();
        try       {
            Thread.sleep(2000);
        } catch (InterruptedException ie)       {
            System.out.println("Thread dont work");       }
        builder.click(subElement).perform();
        try       {
            Assert.assertTrue(driver.findElement(By.xpath("//input[contains(@value, '" + nickname + "')]")).isDisplayed());
            System.out.println("Was logged in");
        } catch (NoSuchElementException nsee)       {
            System.out.println("Wasn't logged in");
            driver.quit();       }    }
    public static void logOut()    {
        Actions builder = new Actions(driver);
        WebElement element = driver.findElement(By.xpath("//li[contains(@id, 'menu-profile')]"));
        WebElement subElement = driver.findElement(By.xpath("//a[contains(@href,'/main/logout/')]"));
        builder.moveToElement(element).perform();
        try       {
            Thread.sleep(2000);
        } catch (InterruptedException ie)       {
```

```
            System.out.println("Thread dont work");        }
        builder.click(subElement).perform();    }
    public static String randomString()    {
        String                                    beforeScramble                                    =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
        String afterScramble = "";
        Random rn = new Random();
        for (int i = 0; i < 25; i++)        {
            afterScramble                    =                    afterScramble                    +
beforeScramble.charAt(rn.nextInt(beforeScramble.length()));        }
        return afterScramble;    }    }
```

```
<?php
namespace application\tests\api\controllers\v1_0;
class UserControllerTest extends \VApiTestCase {
    protected $_apiVersion = 'v1.0';
    public $fixtures=array(
        'users'=>'User',
        'addresses' => 'Address'    );
public function testActionRegister() {
        $response = $this->_doCall('/user/register', false, false, ['email' => '', 'password' => '']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['general'][0],
"EVF_required_params_were_not_passed");
        $response = $this->_doCall('/user/register', false, false, ['email' => 'some', 'password' => '']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['general'][0],
"EVF_required_params_were_not_passed");
        $response = $this->_doCall('/user/register', false, false, ['email' => '', 'password' => 'some']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['general'][0],
"EVF_required_params_were_not_passed");
```

```
        $response = $this->_doCall('/user/register', false, false, ['email' => 'notemail', 'password' => 'somepass']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['email'][0], "Provided email is not valid");
        $response = $this->_doCall('/user/register', false, false, ['email' => 'vova@wasabiventures.com', 'password' => 'somepass']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['email'][0], "Email already exists");
        $email = 'some' . rand(1000, 9999) . '@gmail.com';
        $response = $this->_doCall('/user/register', false, false, ['email' => $email, 'password' => 'somepass', 'fname' => 'Some fname', 'lname' => 'Some lname']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], true);
        $hash = $response['data']['response'];
        $email = 'some' . rand(1000, 9999) . '@gmail.com';
        $response = $this->_doCall('/user/register', false, false, ['email' => $email, 'password' => 'somepass', 'fname' => 'Some fname', 'lname' => 'Some lname', 'image' => 'http://test.jpg/test.jpg']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], true);
        $hash = $response['data']['response'];
        // let's check if hash works
        $response = $this->_doCall('/events/getLive', $hash);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], true);
        $this->assertEquals(count($response['data']['response']), 2);    }
    public function testActoinLogin() {
        $response = $this->_doCall('/user/login', false, ['username' => '', 'password' => '']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['general'][0],
"EVF_required_params_were_not_passed");
        $response = $this->_doCall('/user/login', false, ['username' => 'some', 'password' => '']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
```

```
        $this->assertEquals($response['data']['errors']['general'][0],
"EVF_required_params_were_not_passed");
        $response = $this->_doCall('/user/login', false, ['username' => '', 'password' => 'some']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['general'][0],
"EVF_required_params_were_not_passed");
        $response = $this->_doCall('/user/login', false, ['username' => 'vova@wasabiventures.com',
'password' => 'incorrect']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['password'][0], "Incorrect email or password");
        $response = $this->_doCall('/user/login', false, ['username' => 'vova@wasabiventures.com',
'password' => '111111']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], true);
        $this->assertEquals($response['data']['response']['id'], '1');
        $this->assertEquals($response['data']['response']['auth_token'], 'test1');
        $this->assertEquals($response['data']['response']['fname'], '');
        $this->assertEquals($response['data']['response']['lname'], '');
        $this->assertEquals($response['data']['response']['email'], 'vova@wasabiventures.com');
        $this->assertEquals($response['data']['response']['rewards'], '100000');
        $this->assertEquals($response['data']['response']['image'], '');
        $this->assertEquals($response['data']['response']['is_express'], true);    }
    public function testActionRestorePassword() {
        $response = $this->_doCall('/user/restorePassword', false, false, ['email' => 'notemail']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['email'][0], 'Provided email is not valid');
        $response = $this->_doCall('/user/restorePassword', false, false, ['email' =>
'notexist@mail.com']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['email'][0], 'Email address was not found');
        $response = $this->_doCall('/user/restorePassword', false, false, ['email' =>
'vova@wasabiventures.com']);
```

```
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], true);    }

public function testActionChangePassword() {
        $response = $this->_doCall('/user/changePassword', 'test1', false, ['password' => '', 'password_new' => '']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['general'][0], "EVF_required_params_were_not_passed");
        $response = $this->_doCall('/user/changePassword', 'test1', false, ['password' => '000000', 'password_new' => '222222']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['password'][0], "Incorrect password");
        $response = $this->_doCall('/user/changePassword', 'test1', false, ['password' => '111111', 'password_new' => '222222']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], true);
        $response = $this->_doCall('/user/changePassword', 'test1', false, ['password' => '111111', 'password_new' => '222222']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], false);
        $this->assertEquals($response['data']['errors']['password'][0], "Incorrect password");
        $response = $this->_doCall('/user/changePassword', 'test1', false, ['password' => '222222', 'password_new' => '222222']);
        $this->assertEquals($response['http_code'], '200');
        $this->assertEquals($response['data']['success'], true);    }
```

Додаток Б

Публікація

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 656.222.3

СИСТЕМА МУЛЬТИКРИТЕРІАЛЬНОГО ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ

Березька К.М.[1)], Лабо В.Р.[2)]
Тернопільський національний економічний університет
[1)] к.т.н., доцент; [2)] магістрант

I. Постановка проблеми

У сьогоденні, з розвитком програмного забезпечення, його надійність є гарантією успішної роботи програми. Високу якість функціонування програмного забезпечення можливо отримати завдяки тестуванню як процесу виявлення дефектів. Не надійне програмне забезпечення, що не пройшло процес тестування, може обернутися великими втратами для підприємства чи компанії. Тому і постає потреба розвитку методів тестування програмного забезпечення. Для того, щоб підвищити якість вихідного продукту варто скористатись не одним методом тестування. Саме тому актуальним є побудова цілої системи тестування програмного забезпечення.

II. Мета роботи

Метою даної роботи є підвищення якості програмного забезпечення завдяки створенню системи, що поєднує у собі кілька способів тестування програмного додатку. Дана система повинна показувати більш високі показники надійності та достовірності у порівнянні з існуючими рішеннями.

III. Особливості реалізації програмного комплексу для тестування програмного забезпечення

Проаналізовано найбільш відомі моделі якості програмного забезпечення та зроблено порівняльний аналіз по узагальненому показнику порівняння моделей (рисунок 1).

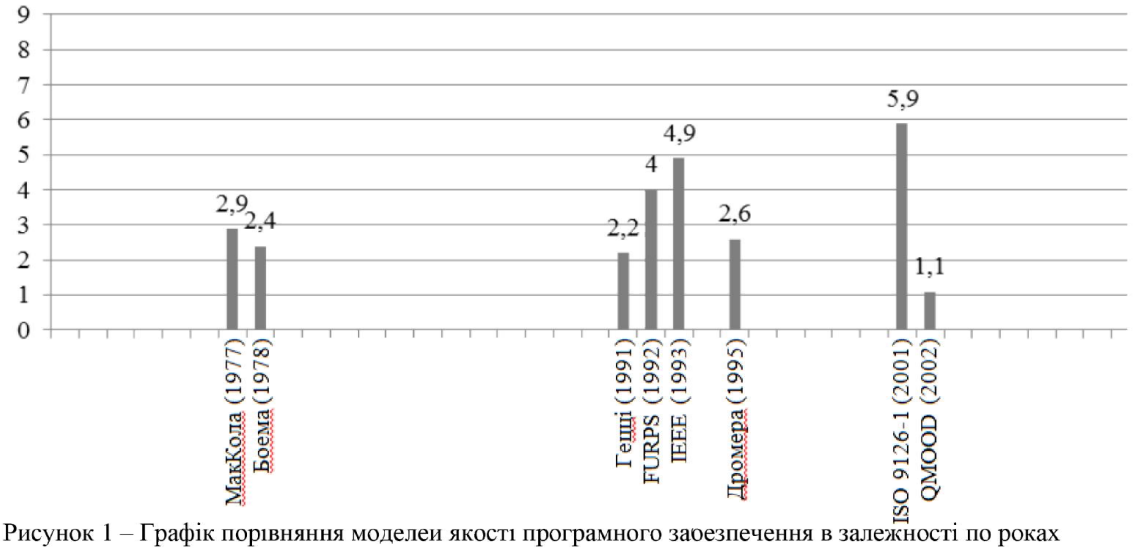

Рисунок 1 – Графік порівняння моделеи якості програмного заоезпечення в залежності по роках

Із рисунку 1 можна зробити висновок, що універсальною є модель ISO 9126-1, її і було обрано як стандарт якості програмного забезпечення за яким буде створено систему мультикритеріального тестування програмного забезпечення (веб-додатків). На рисунку 2 зображено семантичний зміст моделі ISO 9126-1.

Рисунок 2 – Модель якості програмного забезпечення ISO 9126-1

Для створення системи тестування програмного забезпечення за різними критеріями відповідно до обраного стандарту ISO 9126-1 було проаналізовано існуючі методи і відібрано функціональне тестування (за об'єктом тестування), модульне тестування (за рівнем тестування), тестування «білої скриньки» (тестування API) (за знанням системи) та об'єднано в єдину систему.

Для скорочення часу тестування і спрощення його процесу тестування було автоматизовано. Проект реалізовано мовами програмування PHP та Java.

Висновок

Для створення системи мультикритеріального тестування веб-додатків було зроблено аналіз та порівняння найбільш відомих моделей якості програмного забезпечення. Відповідно до критерій моделі якості проаналізовано можливі види тестування. На основі вибраної моделі та видів тестування реалізовано програмну систему для тестування веб-додатків. Особливу увагу було звернено на поєднання різних по критеріях видів тестування.

Список використаних джерел

1. Software engineering. Report on a conference sponsored by the NATO SCIENCE COMMITTEE. Garmisch, Germany, 7th to 11th October 1968 [Електронний ресурс]. – Режим доступу: http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF.
2. Сидоров М.О., Безверха М.А. Якість програмного забезпечення та тестування: Підручник – К.: НАУ, 2010. – 282 с.
3. McCall J.A. Factors in Software Quality / J.A. McCall, P.K. Richards, G.F. Walters // Nat'l Tech.Information Service. – 1977. – Vol. 1, 2, 3.
4. Stefan Wagner. Software product quality control / Stefan Wagner. – Springer, 2013. – 210 p.
5. О.С. Білас. Якість програмного забезпечення та тестування: Навчальний посібник – Львів:Нац. ун-т "Львів. політехніка", 2011. - 214 с.

УДК 658.012

ПРОПОЗИЦІЇ ДО РЕАЛІЗАЦІЇ ПРОЕКТУ СТВОРЕННЯ КОМП'ЮТЕРНОЇ ГРИ ДЛЯ МОБІЛЬНИХ ПЛАТФОРМ

Боднар Є.Л.[1)], Турченко І.В.[2)]
Тернопільський національний економічний університет
[1)]магістрант, [2)]к.т.н.,доцент

І. Постановка проблеми

Світовий ринок ігор обіцяє рости. Аналітики Gartner [1] пророкують йому щорічний приріст в 9 % до кінця 2016 року; в амстердамської Newzoo [2] очікують 7 % темпів зростання [3]. Але ігровий ринок неоднорідний: ігрові консолі та настільні системи займають більшу половину. І тут не так важливі загальні цифри, як те, що відбувається всередині категорій. Платформенні зрушення невідворотні (частка ігор для мобільних платформ на фоні падіння інших тільки зростатиме). Найактивніший приріст спостерігається на ринку ігор для смартфонів і планшетів: в 2013 році - 35 %;

Додаток В

Довідка про використання

Зав. кафедри
комп’ютерної інженерії
д.т.н., проф. О.М. Березькому

ДОВІДКА ПРО ВИКОРИСТАННЯ

Виконана студенткою групи КСМзм-21 факультету комп’ютерних інформаційних технологій Тернопільського національного економічного університету Лабо В.Р. дипломна робота та тему „Система мультикритеріального тестування веб-додатків” відповідає замовленню підприємства, має певну практичну значимість і планується до використання.

Директор підприємства (організації) _____
(підпис)

М.П.