Złożenie pracy online:
2013-09-24 18:25:54
Kod pracy:
10730

Dzmitry Yermakovich
(nr albumu: 17074*INF/LIC)

Praca inżynierska

Serwis zakupów grupowych

Collective purchasing portal

Wydział: Nauk Społecznych i Informatyki

Kierunek: Informatyka

Specjalność: inżynieria oprogramowania

Promotor: dr Henryk Telega

Nowy Sącz, 2013

Streszczenie

Słowa kluczowe

Abstract

Keywords

1

2

Spis treści

Rozdział 1. Cel i zakres pracy

Celem pracy było zaprojektowanie oraz zaimplementowanie serwisu grupowych zakupów z progresywnym systemem zniżek, zintegrowanym z systemem płatności elektronicznej iPay[1]. Docelowo serwis ma być wdrożony na Białorusi, dlatego interfejs użytkownika jest w języku rosyjskim.

Progresywny system zniżek jest wyróżniającą cechą tego serwisu, umożliwia zdobycie największej zniżki na towar lub usługę przy wykupieniu wszystkich wystawionych jednostek, motywując tym samym użytkowników serwisu do dołączania się do akcji promocyjnych oraz do polecania innych osób zainteresowanych tymi akcjami.

W ramach pracy został wykorzystany obiektowy język programowania PHP[2] do generowania stron serwisu po stronie serwera z wykorzystaniem części biblioteki Zend Framework[3]. Do zapewniania interaktywności, sprawdzania formularzy oraz budowania elementów nawigacyjnych został wykorzystany skryptowy język programowania JavaScript z wykorzystaniem biblioteki jQuery[4]. Do gromadzenia informacji użyta została baza danych MySQL[5]. Do komunikacji z bazą danych wykorzystano bibliotekę Doctrine ORM[6], która reprezentuje tabele relacyjnych baz danych w postaci klas modeli oraz udostępnia interfejs do zarządzania danymi za ich pomocą. W przeciwieństwie do innych rozwiązań tego typu, biblioteka potrafi wygenerować modele na podstawie już istniejącej struktury bazy danych. Wygenerowane modele można rozbudowywać o dodatkową funkcjonalność, jakiej potrzebuje programista.

Rozdział 2. Założenia systemu

- Integracja z systemem płatności iPay
- Rejestracja użytkowników z weryfikacją kont mailowych
- Autoryzacja zarejestrowanych użytkowników systemu
- Możliwość odzyskania hasła do panelu użytkownika drogą mailową

1 iPay – system płatności elektronicznej http://www.ipay.by/

2 PHP - http://pl.wikipedia.org/wiki/PHP

3 http://framework.zend.com/

4 jQuery - http://jquery.com/

5 MySQL - http://pl.wikipedia.org/wiki/MySQL

6 Doctrine ORM - http://pl.wikipedia.org/wiki/Doctrine

- Strona startowa z suwakiem, zawierającym wyróżnione akcje promocyjne
- Tworzenie nowych akcji, edycja istniejących akcji bez dołączonych uczestników, wystawianie od nowa akcji zakończonych
- Strona podglądu akcji
- Dołączenie do akcji zarejestrowanych użytkowników po dokonaniu opłaty przez iPay lub przez wprowadzanie kodu promocyjnego
- Przypomnienie o wystawieniu komentarza po 7 dniach po zakończeniu akcji
- Strona kontaktu
- Strona regulaminu serwisu
- Archiwum akcji zakończonych
- System ocen i komentarzy
- Strona użytkownika, zawierająca listy komentarzy otrzymanych oraz wystawionych, statystyki ocen
- Panel użytkownika
  - Zarządzanie danymi osobowymi
  - Zarządzanie treścią wizytówki, tylko dla organizatorów (treść wizytówki jest widoczna na stronie akcji)
  - Zarządzanie treścią regulaminu, tylko dla organizatorów (regulamin jest widoczny na stronie akcji)
  - Zmian hasła dostępu do systemu
  - Lista akcji utworzonych przez użytkownika trwających w danym momencie, dotyczy tylko organizatorów
  - Lista akcji zakończonych, dotyczy tylko organizatorów
  - Lista akcji w których użytkownik bierze udział
  - Lista akcji w których użytkownik brał udział
  - Lista komentarzy do wystawienia
  - Lista komentarzy otrzymanych
  - Lista komentarzy wystawionych
- Komunikacja mailowa
  - Mail przy utworzeniu akcji - do organizatora
  - Mail przy dołączeniu do akcji - do użytkownika dołączającego oraz do

organizatora a także do użytkowników dołączonych już do tej akcji

- Mail po zakończeniu akcji - do organizatora, zawierający listę użytkowników biorących udział w akcji
- Mail do wszystkich dołączonych użytkowników po zakończeniu akcji zawierający w załączeniu certyfikat do wydruku na otrzymanie zniżki

Rozdział 3. Implementacja

3.1. Środowisko programistyczne

Do napisania pracy było wykorzystane zintegrowane środowisko programistyczne NetBeans IDE[7] 7.1.1. Jest to oprogramowanie darmowe i rozpowszechniane na podstawie licencji GNU General Public License[8]. Zawiera niezbędne funkcje do zarządzania większymi projektami, obsługę języków PHP i JavaScript oraz HTML i CSS.

3.2. Ładowanie bibliotek

Na początku każdego skryptu strony jest umieszczany kod, który ładuje skrypt konfiguracyjny, rozpoczyna sesje oraz ładuje bibliotekę Doctrine ORM, która odpowiada za komunikację z bazą danych. Ponieważ protokół HTTP jest bezstanowy dla przechowywania informacji na potrzeby autoryzacji użytkowników lub formularzy są przechowywane w sesji.

```
<?php
require_once('config.php');
Zend_Session::start() ;
Core_System::getInstance()->bootstrap() ;
?>
```

Ilustracja 1: Ładowanie bibliotek

W pliku „config.php” są definiowane zmienne środowiskowe, które później są używane

7 https://netbeans.org/

8 http://pl.wikipedia.org/wiki/GNU_General_Public_License

w całym systemie. Dla automatycznego ładowania klas użyłem klasy „Zend_Loader_Autoloader” z biblioteki Zend Framework, która ładuje plik z klasą na podstawie nazwy klasy. Na potrzeby własnej biblioteki zarejestrowałem przestrzeń nazw „Core”, w której umieściłem singleton „Core_System”, zwierający metodę „bootstrap”, od której zaczyna się ładowanie całego systemu i definiowanie połączenia do bazy. Biblioteka napisana na potrzeby serwisu zawiera ponad 90 klas. Poniżej jest umieszczony kod metody „Core_System::bootstrap”.

```
public function bootstrap()
 {
   // parametry połączenia z bazą danych
   $conString = 'mysql://user_name:password@db_host/db_name' ;
   if( SF_DEVELOPMENT ) {
     $conString = 'mysql://user_name:password@db_host/db_name' ;
   }
   require_once('Doctrine.php') ;
   require_once('Page.php') ;
   // automatyczne ładowanie klas biblioteki Doctrine
   spl_autoload_register( array( 'Doctrine', 'autoload' ) ) ;
   // nazwiązywanie połączenia z bazą
   $manager = Doctrine_Manager::getInstance() ;
   $conn = Doctrine_Manager::connection( $conString, 'doctrine' ) ;
   $conn->setCharset( 'utf8' ) ;
   $conn->setCollate( 'utf8_general_ci' ) ;

   $manager->setAttribute( 'portability', Doctrine::PORTABILITY_ALL ) ;
   $manager->setAttribute( Doctrine::ATTR_AUTOLOAD_TABLE_CLASSES, true ) ;
   $manager->setCharset( 'utf8' ) ;
   $manager->setCollate( 'utf8_general_ci' ) ;

   $conn->setAttribute( Doctrine::ATTR_QUOTE_IDENTIFIER, true ) ;
   // ładowanie modelu bazy danych z kalalogu dbl
   Doctrine::loadModels( 'dbl' ) ;
 }
```

Ilustracja 2: Metoda Core_System::bootstrap

Poniżej umieszczona jest zawartość pliku „config.php”.

```php
<?php
  // ścieżka do katalogu z bibliotekami
  define( 'PATH_LIB', str_ireplace( '\\', '/', dirname( __FILE__ ) ) ) ;
  // ścieżka do głównego katalogu systemu
  define( 'PATH_APP', dirname( PATH_LIB ) ) ;
  // ścieżka do katalogu z własną biblioteką
  define( 'PATH_LIB_CORE', PATH_LIB . '/Core' ) ;

  // dołączenie funkcji pomocniczych
  require_once('include/functions.php') ;
  require_once PATH_LIB . '/Core/functions.php' ;

  set_include_path( implode( PATH_SEPARATOR, array(
    get_include_path(),
    PATH_LIB_CORE,
    PATH_LIB
  ) ) ) ;

  define( 'SF_DEBUG', file_exists( PATH_LIB . '/debug') ) ;

  define( 'SF_DEVELOPMENT', file_exists( PATH_LIB . '/development') ) ;

  define( 'SF_DOMAIN', SF_DEVELOPMENT ? 'selloff' : 'selloff.by' ) ;

  function sf_error_handler( $errno, $errstr, $errfile, $errline ) {
    fdebug( 'Error', $errno, $errstr, $errfile, $errline ) ;
    throw new Exception( 'Error: ' . $errstr ) ;
  }

  if( SF_DEBUG ) {
    // w trybie debugowania pokazuj wszystkie warning'i
    error_reporting( E_ALL^E_NOTICE ) ;
  } else {
    error_reporting( 0 ) ;
  }

  // automatyczne ładowanie klas
  require_once( 'Zend/Loader/Autoloader.php' ) ;
  // rejestrujemy przestrzeń nazw "Core" na potrzeby własnej biblioteki
  Zend_Loader_Autoloader::getInstance()->registerNamespace('Core') ;
?>
```

Ilustracja 3: Zawartość pliku config.php

Napisałem funkcję na potrzeby debugowania, która dodaje wpis do pliku „debug.log” tylko w trybie debugowania. Funkcja ta nie jest skomplikowana, ale jest bardzo użyteczna i zaoszczędza dużo czasu, ma zmienną listę parametrów. Przy zapisywaniu informacji do pliku w zależności od typu parametru wybiera odpowiednią metodę do reprezentacji tekstowej.

Dodaje ona też nazwę klasy i metodę w której została wywołana.

```php
/**
 * Dodaje wpis do pliku debug.log w postaci
 * [ Y-m-d H:i:s ][ $className::$methodName ]: $paramOne | $paramsTwo ...
 *
 * @return void
 */
function fdebug()
{
  if( !SF_DEBUG ) {
    return ;
  }

  $backtrace = debug_backtrace() ;
  $trace = '' ;
  if( count( $backtrace ) > 1 ) {
    $class = trim( $backtrace[ 1 ]['class'] ) ;
    $func = trim( $backtrace[ 1 ]['function'] ) ;
    $trace = sprintf( '[ %s%s ]', !empty( $class ) ? $class . '::' : '', $func )
  }

  $args = func_get_args() ;

  $aux = array() ;
  foreach( $args as $arg ) {
    if( is_string( $arg ) ) {
      $aux[] = $arg ;
    } else {
      $aux[] = var_export( $arg, true ) ;
    }
  }
  $text = sprintf('[ %1$s ]%3$s: %2$s' . PHP_EOL,
    date('Y-m-d H:i:s'), implode( ' | ', $aux ), $trace ) ;
  file_put_contents( SF_LOG , $text, FILE_APPEND ) ;
}
```

Ilustracja 4: Funkcja "fdebug"

3.3. Komunikacja z bazą danych

Do komunikacji z bazą danych użyłem bibliotekę Doctrine ORM[9], która reprezentuje tabele relacyjnych baz danych w postaci klas modeli oraz udostępnia interfejs do zarządzania danymi za ich pomocą. Pierwszy parametr tej metody to jest nazwa katalogu, gdzie ma być umieszczony wygenerowany model, drugi parametr to nazwa połączenia do bazy danych, na podstawie której będzie generowany model, w tym przypadku to jest „doctrine”, to połączenie

9 Doctrine ORM - http://pl.wikipedia.org/wiki/Doctrine

zostało zdefiniowane wcześniej w metodzie „Core_System::bootstrap”. Poniżej został przestawiony kod, który wygeneruje model bazy danych.

```
Doctrine_Core::generateModelsFromDb(
  'dbl_'. date('mdHi') ,
    array( 'doctrine' ),
      array( 'generateTableClasses' => true ) ) ;
```

Ilustracja 5: Generowanie modelu bazy danych

Żeby pokazać jak wyglądają operacje na bazie danych przedstawię klasę „Order” która przechowuje stan zamówienia i jest używania na potrzeby obsługi płatności, które będą przedstawione później. Jak widać klasa Order dziedziczy po wcześniej wygenerowanej klasie „BaseOrder”, która zawiera wszystkie pola oraz powiązania tabeli „Order” jako pola klasy.

Metoda „Order::setPaid” zmienia stan zamówienia na „opłacone”, wywoływana jest w trakcie zapytania od systemu płatności, który potwierdza opłacenie zamówienia. Metoda „Order::setCanceled” zmienia stan zamówienia na „anulowane”.

```
<?php
require_once( 'generated/BaseOrder.php' ) ;

class Order extends BaseOrder
{
  /* ... */

  /**
   * Zmienia stan zamówienia na "opłacone"
   * ( musi być wołana na rozpoczętej transakcji )
   *
   * @throws Core_Exception
   */
  public function setPaid()
  {
    if( $this->status > 0 ) {
      throw new Core_Exception(
        'Не возможно изменить статус заказа на оплачено (status = %s).',
          $this->status ) ;
    }
    $this->status = 1 ;
    $this->save() ;
    $conTmp = $this->getConnectTmp() ;
    $conTmp->status = 1 ;
    $conTmp->save() ;
    $conTmp->So_Product->addConnect( $conTmp ) ;
  }
```

Ilustracja 6: Klasa Order

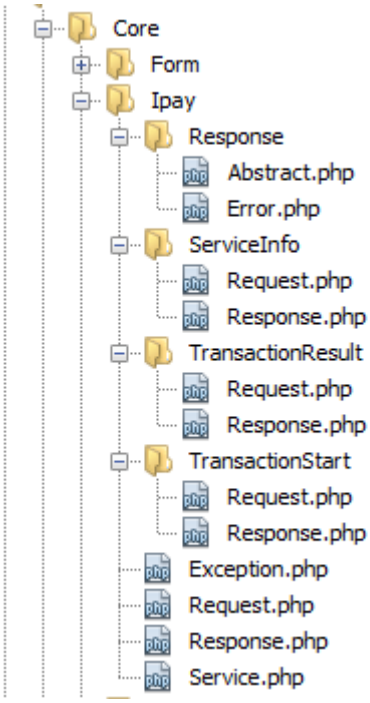
```
32      *
33      * @throws Core_Exception
34      */
35     public function setCanceled()
36     {
37       if( $this->status > 0 ) {
38         throw new Core_Exception(
39           'Не возможно изменить статус заказа на анулировано (status = %s).',
40             $this->status ) ;
41       }
42       $this->status = 2 ;
43       $this->save() ;
44       $conTmp = $this->getConnectTmp() ;
45       $conTmp->status = 2 ;
46       $conTmp->save() ;
47     }
48
49     /**
50      * Ustawia nie opłacone zamówienia jako anulowane
51      *
52      */
53     public static function process()
54     {
55       $time = mktime() ;
56       $time -= Core_System::getInstance()->getOrderLifetimeMin() * 60 ;
57
58       $result = Doctrine::getTable('ConnectTmp')
59         ->createQuery('c')
60         ->innerJoin( 'c.Order o' )
61         ->where( 'c.status = 0' )
62         ->andWhere( 'c.created_at < ?', Core_Db::dateTimeAsSqlText( $time ) )
63         ->execute() ;
64
65       $dbCon = Core_Db::getConnection() ;
66       foreach( $result as $con ) {
67         if( ! $con instanceof ConnectTmp ) {
68           continue ;
69         }
70         try {
71           // nowa transakcja
72           $dbCon->beginTransaction() ;
73           $con->status = 2 ;
74           $con->save() ;
75
76           $order = $con->getOrder() ;
77           if( $order instanceof Order ) {
78             $order->status = 2 ;
79             $order->save() ;
80           }
81           // zatwierdź transakcję
82           $dbCon->commit() ;
83         } catch( Exception $e ) {
84           // wycofaj transakcję w przypadku wystąpienia błędu
85           $dbCon->rollback() ;
86           throw $e ;
87         }
88       }
89     }
90     /* ... */
91   }
```

Ilustracja 7: Klasa Order

W metodzie „Order::process” sprawdzamy wszystkie nie opłacone i nie anulowane zamówienia starsze niż 10 minut, liczba ta jest zwracana przez metodę „Core_System::getOrderLifetimeMin”.

3.4. Obsługa płatności

Na potrzeby obsługi płatności utworzyłem kilka podstawowych klas, z których dziedziczy resztę klas. Na ilustracji poniżej przedstawiona struktura plików, w których zawarte są definicje tych klas.



Ilustracja 8: Klasy do obsługi płatności

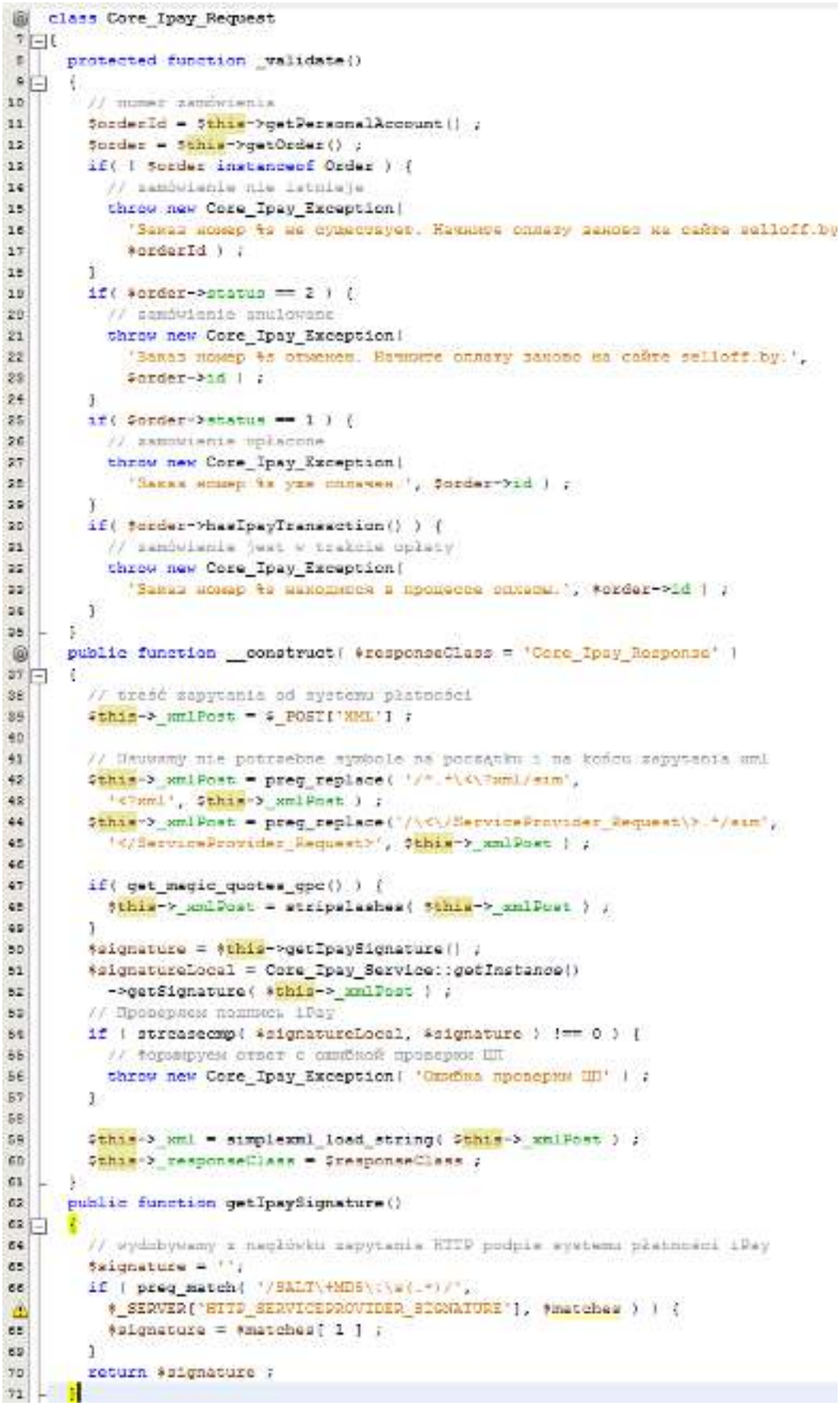
Obsługa płatności wygląda następująco, serwer płatności wysyła zapytanie „ServiceInfo” do serwisu w momencie jak klient rozpoczyna proces opłaty. Wszystkie zapytania są przesłane metodą POST. Treść pojedynczego zapytania umieszczona jest w parametrze o nazwie „XML” w formacie xml. Każde zapytanie zawiera numer zamówienia. W odpowiedzi na zapytanie „ServiceInfo” system zwraca kwotę do zapłaty. Później przychodzi zapytanie „TransactionStart”, które świadczy o rozpoczęciu transakcji opłaty. W tym momencie system zmienia stan zamówienia na „w trakcie opłaty” i zwraca wewnętrzny numer transakcji. Przed przetwarzaniem zapytań „TransactionStart” i „ServiceInfo” system też sprawdza czy zamówienie nie jest już opłacone i czy nie jest w trakcie opłaty, jeśli tak to

zwraca negatywną odpowiedź do systemu płatności. Później następuje trzecie zapytanie od systemu płatności „TransactionResult", które może być pozytywne lub negatywne. Po takim zapytaniu zamówienie przechodzi do stanu opłacone lub anulowane. Poniżej przedstawiłem zawartość singletona „Core_Ipay_Service", który jest punktem wejściowym przy przetwarzaniu zapytań od systemu płatności.

```php
class Core_Ipay_Service
{
  /* ... */
  protected function _sendResponse( Core_Ipay_Response $response )
  {
    $data = $response->getXml() ;
    $signature = $this->getSignature( $data ) ;
    // Tworzymy podpis cyfrowy i wysyłamy do systemu płatności
    header( 'Content-Type: text/xml; charset=windows-1251' ) ;
    header( 'Status: 200 OK' ) ;
    header( "ServiceProvider-Signature: SAIT+MD5: " . $signature ) ;
    echo $data ;
  }

  public function process()
  {
    $con = Core_Db::getConnection() ;
    try {
      $con->beginTransaction() ;

      $request = new Core_Ipay_Request() ;
      $type = $request->getRequestType() ;

      // na podstawie typu zapytania 'ServiceInfo', 'TransactionStart'
      // lub 'TransactionResult' tworzymy odpowiedni obiekt odpowiedzi
      $requestClass = 'Core_Ipay_' . ucfirst( $type ) . '_Request' ;
      $request = new $requestClass() ;

      $this->_sendResponse( $request->getResponse() ) ;

      $con->commit() ;
    } catch( Exception $e ) {
      $con->rollback() ;
      $message = $e->getMessage() ;
      if( ! $e instanceof Core_Ipay_Exception ) {
        $message = 'Произошла непредвиденная ошибка.' ;
      }
      $error = new Core_Ipay_Response_Error( $message ) ;
      $this->_sendResponse( $error ) ;
    }
  }
  /* ... */

  public function getSalt()
  {
    return addslashes( 'eS1456a2fx>' ) ;
  }

  public function getSignature( $data )
  {
    return md5( $this->getSalt() . $data ) ;
  }
}
```

Ilustracja 9: Klasa Core_Ipay_Service

Podstawową klasą dla obsługi zapytań od systemu płatności jest klasa „Core_Ipay_Request", zawiera ona głównie metody do parsowania zapytania i sprawdzania podpisu zapytania oraz utworzenie obiektu odpowiedzi. Poniżej przedstawiłem główne metody tej klasy.



```
class Core_Ipay_Request
{
  protected function _validate()
  {
    // номер заказа
    $orderId = $this->getPersonalAccount() ;
    $order = $this->getOrder() ;
    if( ! $order instanceof Order ) {
      // заказ не существует. Начните оплату заново на сайте selloff.by
      throw new Core_Ipay_Exception(
        'Заказ номер %s не существует. Начните оплату заново на сайте selloff.by',
        $orderId ) ;
    }
    if( $order->status == 2 ) {
      // заказ аннулирован
      throw new Core_Ipay_Exception(
        'Заказ номер %s отменен. Начните оплату заново на сайте selloff.by.',
        $order->id ) ;
    }
    if( $order->status == 1 ) {
      // заказ оплачен
      throw new Core_Ipay_Exception(
        'Заказ номер %s уже оплачен.', $order->id ) ;
    }
    if( $order->hasIpayTransaction() ) {
      // заказ в процессе оплаты
      throw new Core_Ipay_Exception(
        'Заказ номер %s находится в процессе оплаты.', $order->id ) ;
    }
  }
  public function __construct( $responseClass = 'Core_Ipay_Response' )
  {
    // тело запроса от системы платежей
    $this->_xmlPost = $_POST['XML'] ;

    // Удаляем не нужные символы в начале и в конце запроса xml
    $this->_xmlPost = preg_replace( '/^.*?(<\?xml.*$/sm',
      '<?xml', $this->_xmlPost ) ;
    $this->_xmlPost = preg_replace('/(<\/ServiceProvider_Request>).*?$/sm',
      '</ServiceProvider_Request>', $this->_xmlPost ) ;

    if( get_magic_quotes_gpc() ) {
      $this->_xmlPost = stripslashes( $this->_xmlPost ) ;
    }
    $signature = $this->getIpaySignature() ;
    $signatureLocal = Core_Ipay_Service::getInstance()
      ->getSignature( $this->_xmlPost ) ;
    // Проверяем подпись iPay
    if ( strcasecmp( $signatureLocal, $signature ) !== 0 ) {
      // формируем ответ с ошибкой проверки ЭП
      throw new Core_Ipay_Exception( 'Ошибка проверки ЭП' ) ;
    }

    $this->_xml = simplexml_load_string( $this->_xmlPost ) ;
    $this->_responseClass = $responseClass ;
  }
  public function getIpaySignature()
  {
    // извлекаем из заголовка запроса HTTP подпись системы платежей iPay
    $signature = '';
    if ( preg_match( '/SALT\+MD5\:\w(.*)/',
      $_SERVER['HTTP_SERVICEPROVIDER_SIGNATURE'], $matches ) ) {
      $signature = $matches[ 1 ] ;
    }
    return $signature ;
  }
```

Ilustracja 10: Klasa Core_Ipay_Request

Rozdział 4. Opis działania

4.1. Strona główna

Ilustracja 11: Strona główna

Strona główna systemu prezentuje suwak, w którym zawarte są dwa slajdy przedstawiające w formie graficznej funkcjonalność całego serwisu oraz pozostałe slajdy wyświetlające ostatnio utworzone akcje.

Dzięki takiemu rozwiązaniu potencjalny użytkownik jest w stanie w bardzo krótkim czasie dotrzeć do najnowszych promocji. Po kliknięciu w akcję przedstawioną w suwaku użytkownik przechodzi na podstronę podglądu akcji gdzie może przystąpić do dołączenia do akcji.

Ilustracja 12: Najnowsze akcje promocyjne na stronie głównej

4.2. Strona podglądu akcji

Strona podglądu akcji jest jedną z najważniejszych, ponieważ prezentuje wszystkie szczegółowe dane na temat oferowanego produktu bądź usługi. Podgląd akcji zawiera również takie elementy jak: galerię zdjęć, informację o aktualnej cenie, aktualnym rabacie, możliwej do osiągnięcia minimalnej cenie oraz możliwym do osiągnięcia maksymalnym rabacie.

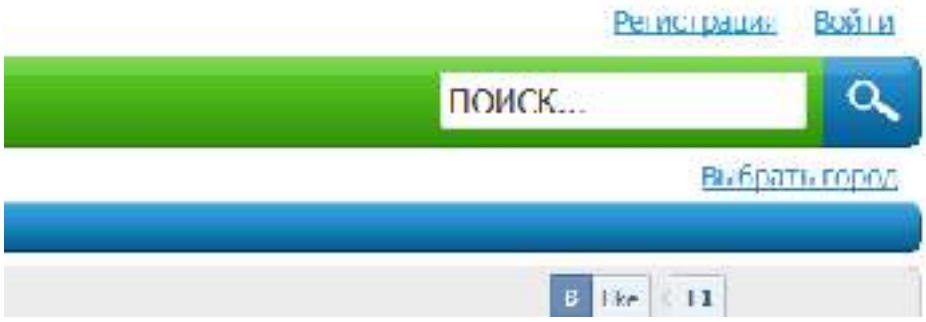
Oprócz powyższych przedstawia również dokładny opis produktu lub usługi, informację o sposobach dostawy i płatności, listę osób które dołączyły już do akcji a także komentarze, które za pośrednictwem API powiązane są z portalem społecznościowym vk.com. Z poziomu tej podstrony każdy użytkownik ma również możliwość zgłoszenia naruszenia zasad przez organizatora akcji.

Ilustracja 13: Strona podglądu akcji

4.3. Rejestracja

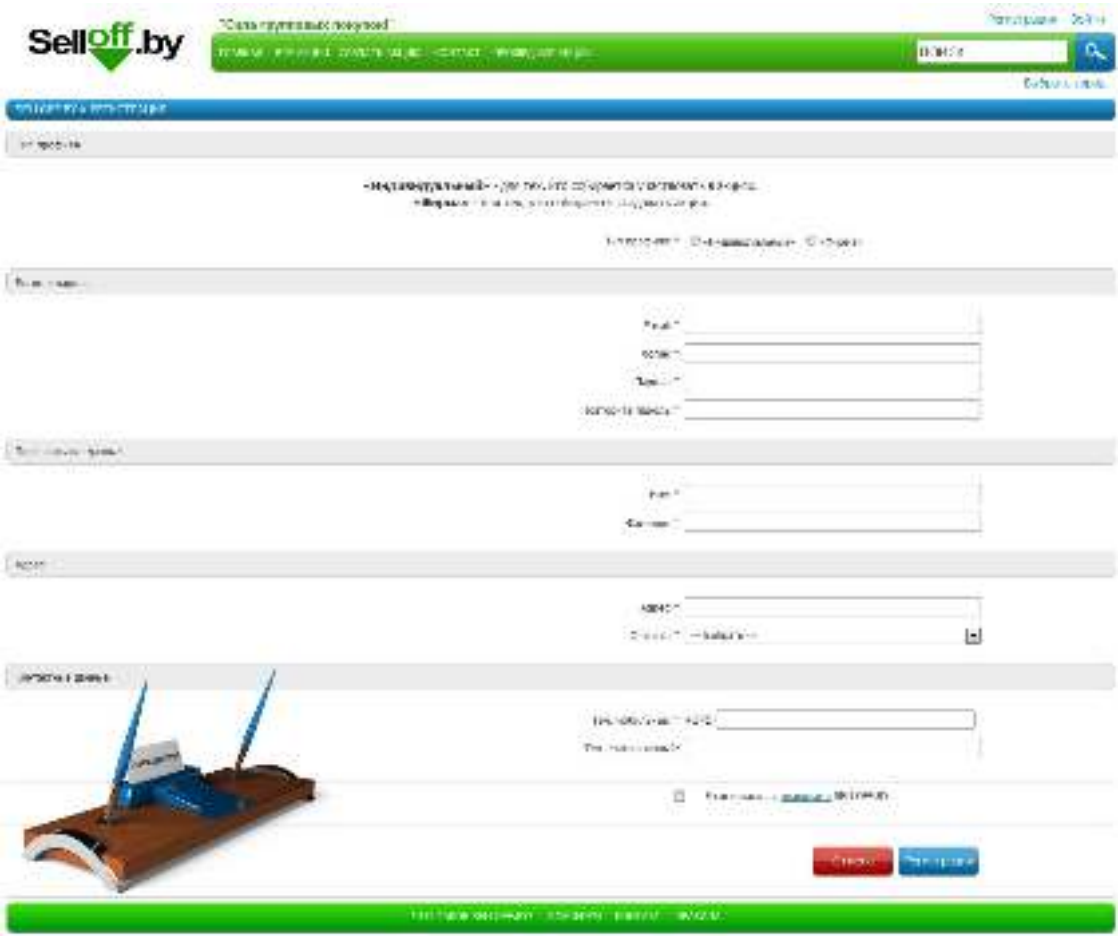
Aby użytkownik mógł przystąpić do akcji i uzyskać certyfikat uprawniający go do zakupu z rabatem musi przejść przez proces rejestracji. Rejestracja możliwa jest z poziomu

każdej podstrony w prawym górnym rogu.



Ilustracja 14: Odsyłacz do rejestracji w serwisie

Po przejściu na tę podstronę i prawidłowym wypełnieniu wymaganych pól formularza użytkownik zostaje zarejestrowany w systemie. Podczas rejestracji użytkownik musi wybrać typ konta. Konto „zwykłe" bądź konto „firmowe". Konto zwykłe umożliwia jedynie dokonywanie zakupów a konto firmowe dodatkowo tworzenie własnych akcji. Po rejestracji na podany w formularzu adres zostaje wysłany e-mail z dynamicznie wygenerowanym odsyłaczem aktywacyjnym.



Ilustracja 15: Strona rejestracji

Taka procedura ma za zadanie weryfikację czy osoba rejestrująca podała swój adres email. Po odebraniu wiadomości i kliknięciu w odsyłacz użytkownik zostaje przeniesiony na podstronę z informacją o pozytywnym zakończeniu rejestracji oraz zostaje poinformowany o możliwości zalogowania się do serwisu.

4.4. Logowanie

Logowanie odbywa się z poziomu każdej podstrony serwisu poprzez kliknięcie odsyłacza „zaloguj”. W wyświetlonym okienku użytkownik musi wprowadzić adres e-mail, podany przy rejestracji oraz hasło.

Ilustracja 16: Okno logowania

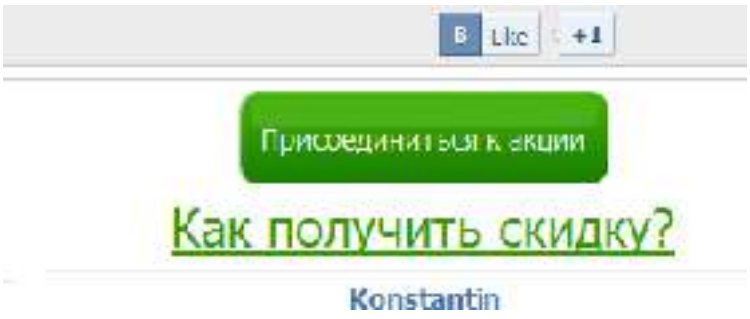
W przypadku, gdyby użytkownik zapomniał swojego hasła, zawsze ma możliwość jego zresetowania poprzez funkcję „wygeneruj nowe hasło” znajdującego się pod formularzem logowania.

Ilustracja 17: Okno generowania nowego hasła

Dla tego, że hasła użytkowników są jednostronnie szyfrowane, nie ma możliwości ich odzyskania a jedynie istnieje możliwość zresetowania istniejącego. Po wprowadzeniu adresu e-mail system sprawdza czy podany adres jest zarejestrowany w systemie, jeśli tak to jest generowane nowe hasło i wysyłane na podany wcześniej e-mail.

4.5. Dołączenie do akcji

Po zalogowaniu użytkownik może dołączać się do akcji z poziomu opisanej już wcześniej podstrony podglądu akcji. Dołączenie odbywa się poprzez wciśnięcie przycisku „Dołącz do akcji”.



Ilustracja 18: Przycisk dołączenia do akcji

Wciśnięcie tego przycisku powoduje wyświetlenie okna z formularzem dołączenia do akcji.

Ilustracja 19: Parametry dołączenia do akcji

Po wybraniu liczby sztuk, sposobu opłaty oraz sposobu dostarczania użytkownik dostaje obliczoną aktualną cenę na towar lub usługę oraz cenę za dołączenie do akcji. Po potwierdzeniu wybranych parametrów użytkownik jest przenoszony do formularza dokonania opłaty za dołączenie do akcji.

Na rozpoczęcie dokonania opłaty użytkownik ma 10 minut, w tym czasie wybrana ilość towaru jest zarezerwowana. Jeśli użytkownik nie dokona opłaty w podanym czasie lub system

płatności zwróci anulowanie transakcji opłaty to rezerwacja jest anulowana. Dokonać opłatę za dołączenie do akcji użytkownik może na trzy sposoby:

- Przez wysłanie sms'a na określony numer z określoną treścią
- Przez dokonanie opłaty na stronie systemu płatności
- Przez wprowadzenie kodu promocyjnego, który użytkownik może dostać od serwisu jako prezent

W momencie kiedy system stwierdzi że dołączenie do akcji zostało opłacone, użytkownik zostaje dołączony do akcji. Na potwierdzenie dołączenia do akcji użytkownik dostaje e-mail. E-mail otrzymuje też organizator akcji oraz użytkownicy które do tej pory dołączyli się do akcji.

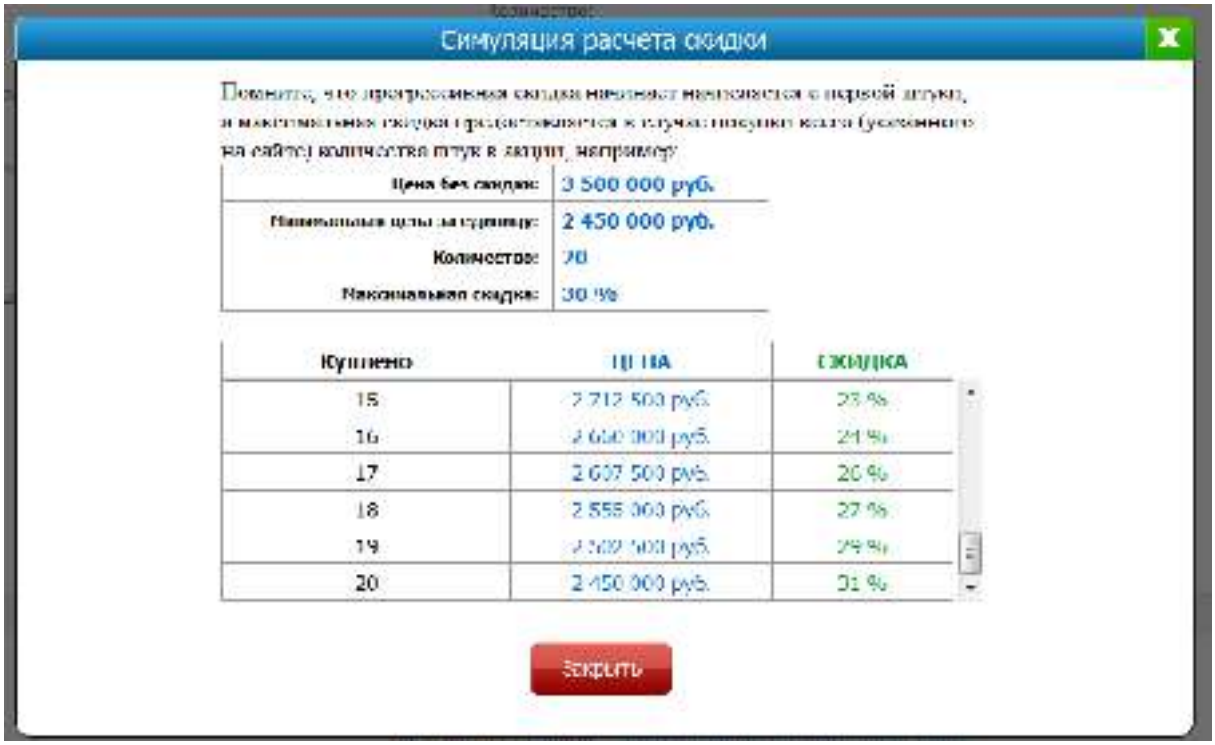
Ilustracja 20: Dokonanie opłaty za dołączenie

4.6. Tworzenie akcji

Jedną z najważniejszych stron serwisu jest strona tworzenia akcji. Przy tworzeniu akcji organizator zobligowany jest wypełnić wszystkie wymagane pola formularza, podając nazwę produktu lub usługi, wybierając kategorię produktową, określając liczbę dostępnych sztuk,

kompletów bądź par, limit na zakup – co oznacza, że organizator może zastrzec, że jeden użytkownik ma możliwość zakupu tylko jednej sztuki lub usługi.

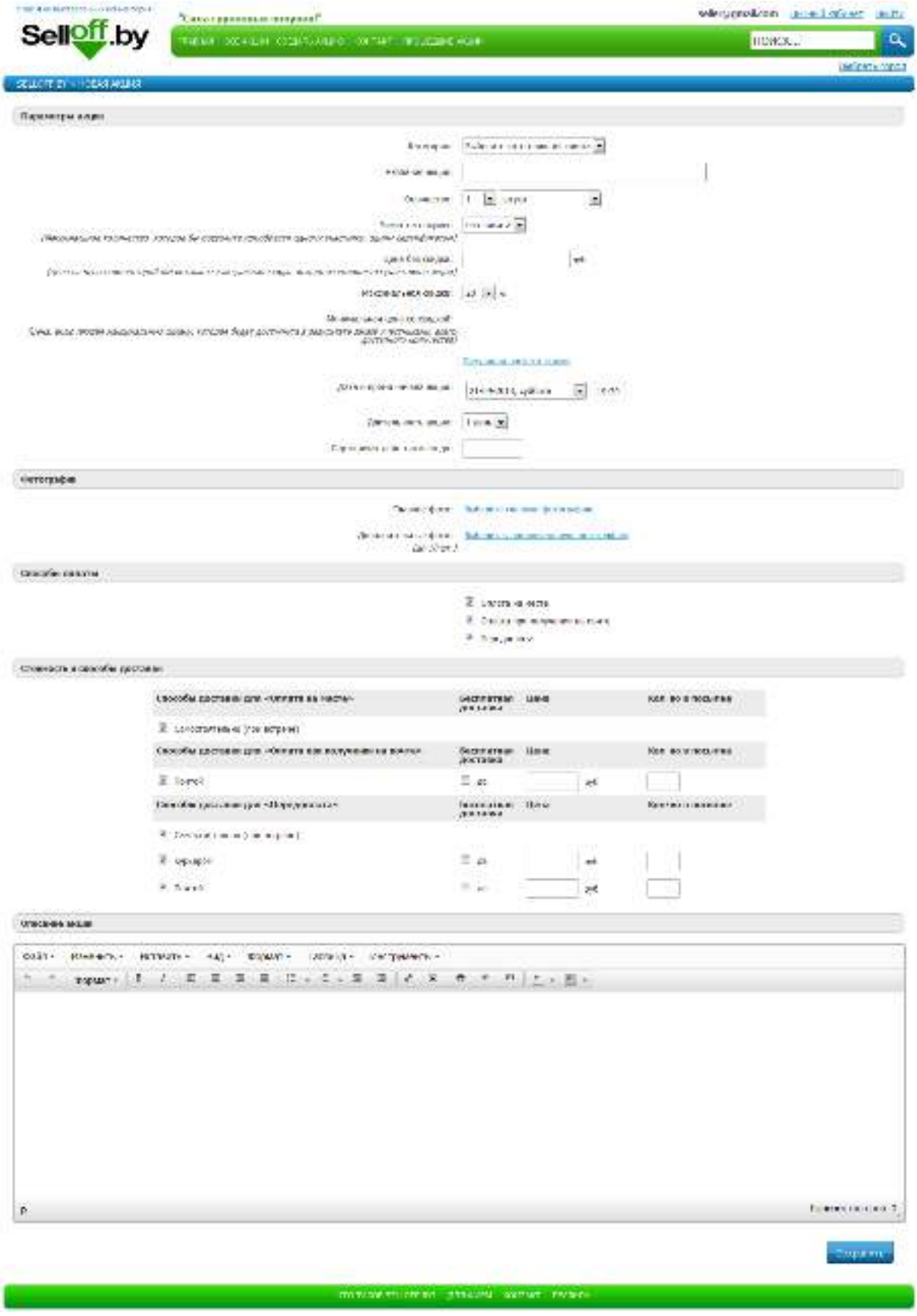
Podanie ceny przed rabatem umożliwi użytkownikom porównanie aktualnej zniżki do ceny rynkowej danego produktu. Zaznaczenie odpowiedniej zniżki spowoduje po zakończeniu wypełniania formularza stworzenie przez system tabeli zniżek progresywnych wizualizując tym samym jak będzie malała cena wraz ze wzrostem zainteresowania produktem. Poniżej przedstawiony jest wygląd strony ze schematem zniżek.



Ilustracja 21: Strona zniżek progresywnych

Organizator dodatkowo ma możliwość ustalenia daty i godziny rozpoczęcia akcji, dzięki czemu może zaplanować sobie całą kampanię reklamową na wybrany przez siebie dzień. Ustalenie daty spowoduje, że to system będzie pamiętał kiedy rozpocząć wyświetlanie akcji a organizator w tym czasie będzie mógł się zająć innymi rzeczami. Formularz udostępnia również opcję, w której organizator ustala czas trwania akcji w zakresie od 1 do 7 dni. W tym czasie akcja będzie wyświetlona w systemie. Natomiast określenie daty ważności certyfikatu umożliwia tak jakby przedłużenie trwania promocji od czasu zakończenia wyświetlania ogłoszenia do czasu wygaśnięcia ważności certyfikatu.

Organizator ma możliwość umieszczania zdjęć produktów w celu efektowniejszego przedstawienia swojej oferty. Umieszczenie ciekawego zdjęcia przyciągnie wzrok potencjalnych klientów. W formularzu przewidziane jest miejsce na dodanie aż do dziesięciu zdjęć. Podczas tworzenia akcji organizator określa dostępne rodzaje płatności oraz transportu. Okno edytora tekstu widoczne w dolnej części formularza służy do wprowadzania opisu akcji.

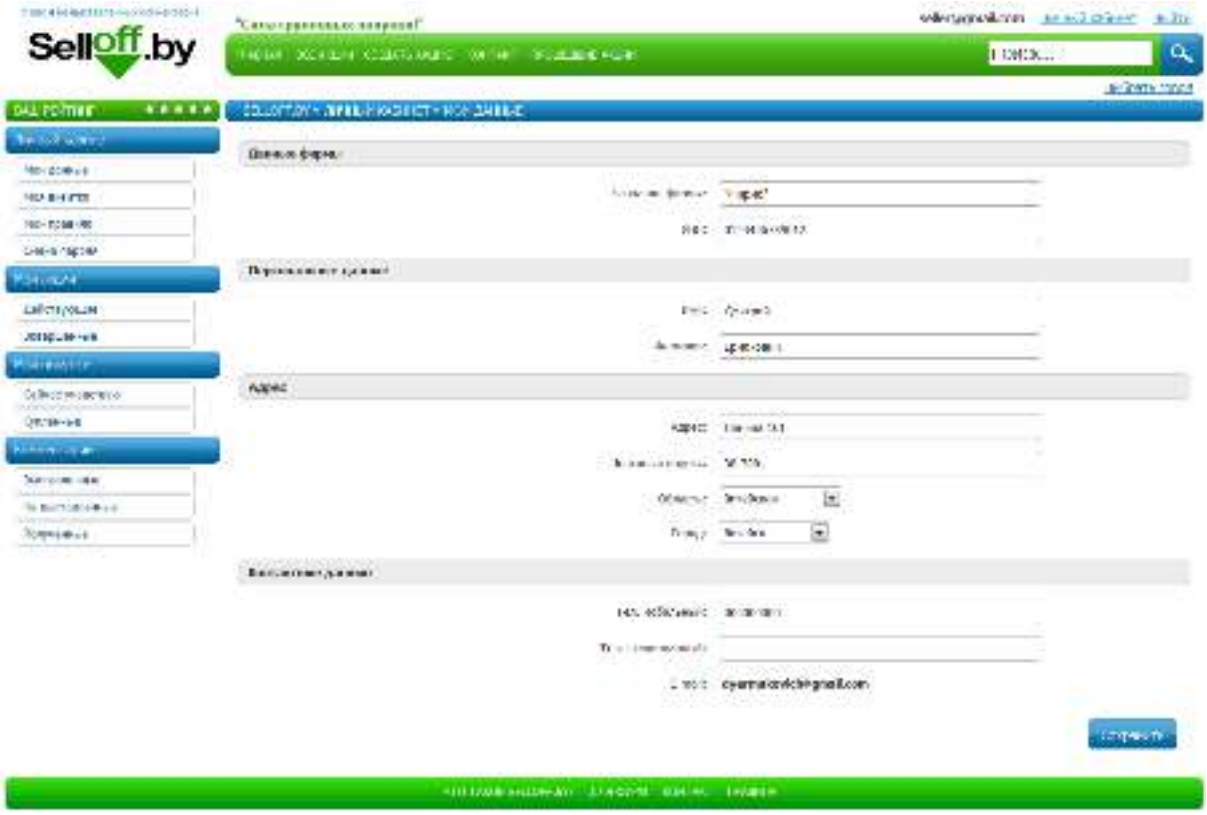


Ilustracja 22: Strona tworzenia akcji

4.7. Panel użytkownika

Po zalogowaniu użytkownik ma dostęp do panelu użytkownika, który w zależności od rodzaju konta udostępnia odpowiednie funkcje.

Każdy użytkownik ma dostęp do strony edycji danych osobowych. Ta strona ma na celu umożliwienie zmiany niektórych danych osobowych podanych przy rejestracji.



Ilustracja 23: Strona edycji danych osobowych

Strony „Moja wizytówka” oraz „Mój regulamin” są dostępne tylko dla użytkowników kont firmowych. Te strony umożliwiają dodanie opisu profilu działalności firmy a także dodania własnego regulaminu. Wprowadzenie tych opisów wzbogaci oferty organizatorów ponieważ właśnie te opisy będą prezentowane wraz z produktami danego organizatora na podstronie podglądu akcji.

W panelu użytkownik też ma dostęp do podstron zawierających:

- akcje do których dołączył i są w trakcie trwania
- akcje do których dołączył i już są zakończone
- akcje które wystawił i jeszcze trwają (tylko konto firmowe)
- akcje które wystawił i już są zakończone (tylko konto firmowe)
- komentarze do wystawienia
- komentarze wystawione
- komentarze otrzymane

4.8. System komentarzy i ocen

Serwis wyposażony jest w system komentarzy i ocen. Każdy użytkownik po dokonaniu transakcji zakupu lub sprzedaży, ma możliwość wystawienia komentarza swojemu kontrahentowi. Inteligencja systemu polega na tym, że system czuwa nad przypomnieniem o wystawieniu komentarza i oceny. Po zakończonej transakcji system przesyła przypomnienie wraz z bezpośrednim odsyłaczem do systemu oceny z zakodowanym ciągiem znaków identyfikującym danego użytkownika. Odsyłacz taki umożliwia autoryzację użytkownika bez zmuszania go do logowania się do systemu.

Ilustracja 24: System komentarzy i ocen

Po zalogowaniu użytkownik, który ma do wystawiania komentarze widzi komunikat u góry strony, którego kliknięcie powoduje przejście do strony wystawiania komentarzy. Jeśli użytkownik nie wystawił komentarza po 7 dniach po zakończeniu akcji, system wysyła jednorazowe przypomnienie o wystawieniu komentarza.

Ilustracja 25: Komunikat o nie wystawionych komentarzach

W wyniku działania systemu komentarzy na stronie każdego użytkownika widoczna jest tabela ocen wystawionych w ciągu ostatnich 7 dni, 30 dni i za cały okres istnienia użytkownika w systemie.

Ilustracja 26: Strona użytkownika

Rozdział 5. Zakończenie

W trakcie pracy zdobyłem doświadczenie w projektowaniu oraz implementacji serwisu internetowego. Poszerzyłem wiedzę w projektowaniu bazy danych. Otrzymałem doświadczenie w programowaniu obiektowym. Dużo czasu spędziłem nad projektowaniem interfejsu użytkownika. Dostałem doświadczenie w integracji serwisu z zewnętrznym systemem płatności.

W wyniku danej pracy powstał wielofunkcyjny i innowacyjny serwis internetowy „selloff.by". Nazwa serwisu, w tłumaczeniu z angielskiego sugeruje zniżki oraz wyprzedaży co doskonale odzwierciedla zawartość, funkcjonalność oraz podstawowy cel danej strony internetowej. Pomysł na dany serwis jest przemyślany i wiązy się z dużym zainteresowaniem internautów rożnego rodzaju akcjami oraz zakupów grupowych. Główną ideą jest system zniżek progresywnych który odzwierciedla powstały wielu lat temu system zakupów grupowych oraz sprzedaż hurtową gdzie liczba potencjalnych kupujących decyduje o cenę oraz warunki zakupu. Oprócz tego system pozwala na intensywny marketing internetowy dla firm i przedsiębiorstw. Poprzez zniżkę progresywną firma daje tylko relewantną zniżkę co sprawia że każda akcja jest udana a nakłady są proporcjonalne liczbę uczestników.