



**POLITECHNIKA  
RZESZOWSKA**  
im. IGNACEGO ŁUKASIEWICZA

## **Aplikacje Internetowe**

Stowarzyszeniowa Wypożyczalnia Gier i Książek:

Moduł zarządzania wypożyczalnią

Przytuła Norbert

EF/C-DU-AI-L7

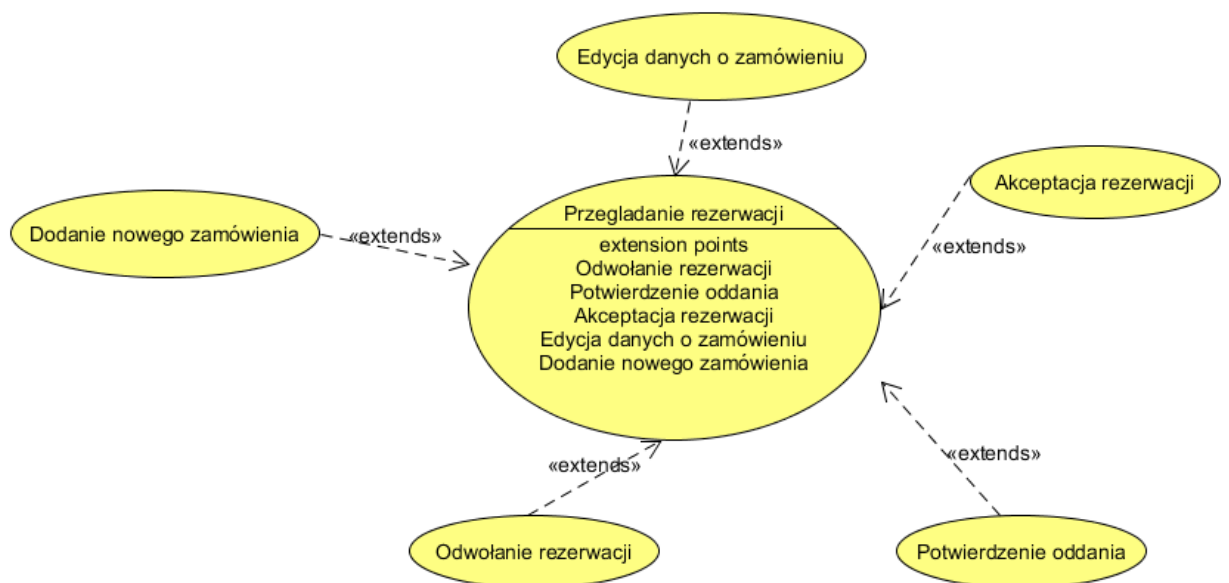
127633

## 1. Opis modułu 'Zamowienia'

Moduł 'Zamowienia' ma za zadanie zarządzanie rezerwacjami przedmiotów. Jego funkcją jest umożliwienie użytkownikom przeglądania informacji na temat złożonych na nich zamówień oraz ich stanu. Ponadto, w przypadku administratorów systemu, moduł ma umożliwić wykonywanie podstawowych operacji na zamówieniach znajdujących się w bazie danych (takie jak zmiana statusu zamówienia, edycja kodu przedmiotu lub osoby wypożyczającej). Dostępu do modułu nie posiadają użytkownicy niezalogowani.

## 2. Diagram przypadków użycia (UML)

Przypadki użycia dla danego modułu opisuje rysunek poniżej:



Rysunek 1: Diagram przypadków użycia modułu

### 3. Diagram związków encji (ERD)

Moduł opiera się na tabeli 'zamówienia', której kolumny są zdefiniowane następująco:

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Dodatkowo
1	id_zamowienia	int(5)			Nie	Brak	AUTO_INCREMENT
2	id_uzytkownika	varchar(255)			Nie	Brak	
3	id_przedmiotu	varchar(255)			Nie	Brak	
4	data	date			Tak	NULL	
5	informacje	varchar(255)			Tak	NULL	
6	status	varchar(255)			Nie	Brak	

Rysunek 2: Tabela 'zamowienia' w MySQL

Tabela posiada 2 klucze obce, gdzie:

- id\_uzytkownika - jest to nazwa użytkownika, znajdująca się w tabeli 'uzytkownik' w kolumnie 'username',
- id\_przedmiotu – jest to kod przedmiotu, znajdujący się w tabeli 'skladzik' w kolumnie 'kod'.

### 4. Dane technologiczne:

Technologia: PHP

Środowisko: NetBeans

Framework: Zend Framework 2

Dane repozytorium:

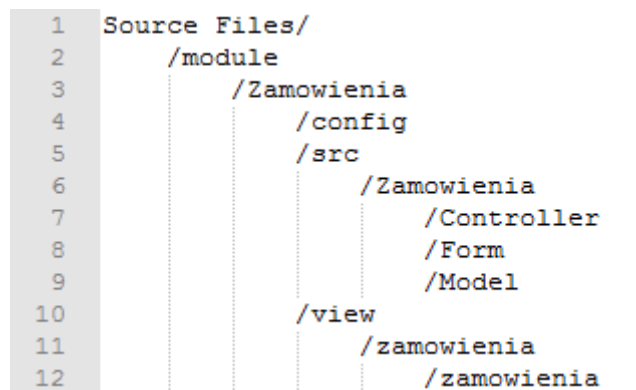
- Serwis hostingowy: GitHub
- Nazwa użytkownika: OSCypek
- Nazwa repozytorium: zskcs-skladzik
- Adres URL: <https://github.com/OSCypek/zskcs-skladzik>

### 5. Proces tworzenia modułu

#### a. Stworzenie drzewa modułu

Pierwszym krokiem, jaki należy wykonać, jest stworzenie odpowiedniej struktury folderów i plików. Dzięki temu program będzie mógł w prosty sposób odwoływać się

do potrzebnych metod bez konieczności szczegółowego deklarowania ścieżek dostępu do plików. Podstawowe drzewo przedstawia rysunek poniżej:



Rysunek 3: Struktura projektu

## b. Stworzenie plików konfiguracyjnych modułu

Kolejnym krokiem jest przygotowanie pliku *Module.php*, który będzie automatycznie konfigurował ścieżki dostępu do zasobów modułu. Ponadto, plik ma za zadanie odpowiednio odebrać wyniki z bazy danych, zdefiniowanych w pliku *ZamowieniaTable*. Następnie należy stworzyć plik *module.config.php* w folderze **config**, którego zadaniem jest zdefiniowanie kontrolera modułu oraz jego akcji.

```
class Module implements AutoloaderProviderInterface, ConfigProviderInterface
{
    public function getAutoloaderConfig()
    {
        return array('Zend\Loader\ClassMapAutoloader' => array(__DIR__ . '/autoload_classmap.php',),
                    'Zend\Loader\StandardAutoloader' =>
                        array('namespaces' => array(__NAMESPACE__ => __DIR__ . '/src/' . __NAMESPACE__,),),);
    }

    public function getConfig()
    {return include __DIR__ . '/config/module.config.php';}

    public function getServiceConfig()
    {
```

Rysunek 4: Fragment kodu *Module.php*

## c. Stworzenie plików encji

Następną rzeczą, jaką należy wykonać, jest przygotowanie plików *Zamowienia.php* oraz *ZamowieniaTable.php* w folderze **src/Model**. Pierwszy plik definiuje kolumny encji, które będziemy wykorzystywać w module, zawiera również odpowiednie filtry dla każdej kolumny (np. czy dane pole jest wymagane przy wypełnianiu arkusza).

```

public function exchangeArray($data)
{
    $this->id_zamowienia = (isset($data['id_zamowienia'])) ? $data['id_zamowienia'] : null;
    $this->id_uzytkownika = (isset($data['id_uzytkownika'])) ? $data['id_uzytkownika'] : null;
    $this->id_przedmiotu = (isset($data['id_przedmiotu'])) ? $data['id_przedmiotu'] : null;
    $this->data = (isset($data['data'])) ? $data['data'] : null;
    $this->informacje = (isset($data['informacje'])) ? $data['informacje'] : null;
    $this->status = (isset($data['status'])) ? $data['status'] : null;
}

public function setInputFilter(InputFilterInterface $inputFilter)
{
    throw new \Exception("Not used");
}

public function getArrayCopy()
{
    return get_object_vars($this);
}

public function getInputFilter()
{
    if (!$this->inputFilter) {
        $inputFilter = new InputFilter();

        $inputFilter->add(array(
            'name' => 'id_zamowienia',
            'required' => true,

```

Rysunek 5: Fragment pliku 'Zamowienia.php'

Plik *ZamowieniaTable.php* definiuje operacje wykonywane na bazie danych, takie jak pobranie danego wiersza, kilku wierszy spełniających warunek (SELECT ... WHERE...;) oraz zapisanie danych do odpowiedniego wiersza (saveZamowienia).

```

public function fetchAll_archives()
{
    $resultSet = $this->tableGateway->select(array('status' => 'Archiwalne'));
    return $resultSet;
}

public function getZamowienia($id)
{
    $id_zamowienia = (int) $id;
    $rowset = $this->tableGateway->select(array('id_zamowienia' => $id_zamowienia));
    $row = $rowset->current();
    if (!$row) {
        throw new \Exception("Could not find row $id_zamowienia");
    }
    return $row;
}

public function saveZamowienia(Zamowienia $zamowienia)
{
    $data = array(
        //'data_dodania' => $article->data_dodania,
        'data' => date("Y-m-d H:i:s",mktime()),
        'id_uzytkownika' => $zamowienia->id_uzytkownika,
        'id_przedmiotu' => $zamowienia->id_przedmiotu,
        'status' => $zamowienia->status,
        'informacje' => $zamowienia->informacje,

```

Rysunek 6: Fragment pliku 'ZamowieniaTable.php'

## d. Stworzenie abstrakcyjnego kontrolera

Po wykonaniu wcześniejszych kroków, należy zdefiniować plik *ZamowieniaController.php* w folderze **src/Controller**. Plik odpowiedzialny jest za zarządzanie modułem oraz jego odpowiednimi akcjami, a także inicjalizuje tworzenie nowych widoków.

```

public function editAction()
{
    $id_zamowienia = (int) $this->params()->fromRoute('id_zamowienia', 0);
    if (!$id_zamowienia) {
        return $this->redirect()->toRoute('zamowienia', array(
            'action' => 'add' ));}
    try {$zamowienia = $this->getZamowieniaTable()->getZamowienia($id_zamowienia);}
    catch (\Exception $ex) {
        return $this->redirect()->toRoute('zamowienia', array(
            'action' => 'index'));}
    $form = new ZamowieniaForm();
    $form->bind($zamowienia);
}

```

Rysunek 7: Fragment pliku 'ZamowieniaController.php'

## e. Określenie formy wyświetlanych formularzy

Kolejną czynnością jest stworzenie pliku *ZamowieniaForm.php* w folderze **src/Form**. Plik ma za zadanie określenie funkcjonalności każdej kolumny encji podczas wykonywania operacji dodawania zamówienia, edytowania. Definiuje takie parametry, jak

- typ kolumny (integer, text, itp.),
- rodzaj uzupełniania pola (pole wpisywania, dropbox, data),
- nazwę wyświetlaną kolumny.

```

$this->add(array(
    'name' => 'informacje',
    'type' => 'Text',
    'options' => array(
        'label' => ' ', ), ));
$this->add(array(
    'name' => 'status',
    //'type' => 'Text',
    'type' => 'Zend\Form\Element\Select',
    'options' => array(
        'label' => ' ',
        'value_options' => array(
            'Oczekujące' => 'Oczekujące',
            'Zaakceptowane' => 'Zaakceptowane',
            'Archiwalne' => 'Archiwalne',
        ), ),
));

```

Rysunek 8: Fragment pliku 'ZamowieniaForm.php'

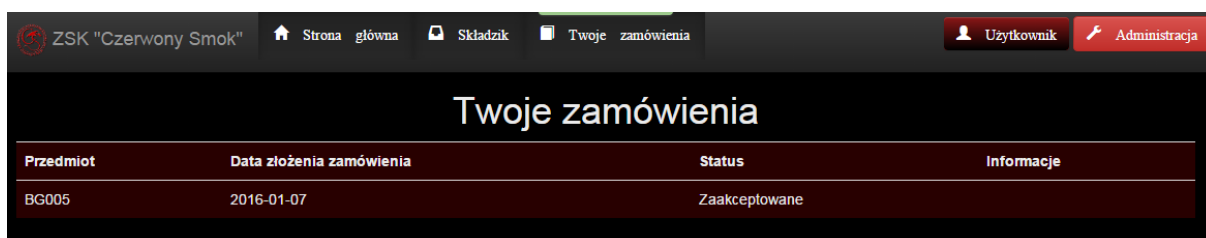
## f. Utworzenie widoków

Następnym krokiem jest utworzenie wszystkich widoków, które są wymagane przez moduł (jednocześnie które odwołują się do odpowiednich metod zawartych w pliku kontrolera). Wszelkie stylistyczne operacje należy zadeklarować w plikach *style.css* w folderze **public** oraz w pliku *layout.phtml* w głównym module 'Application'. W przypadku modułu 'Zamowienia' zawierają się następujące widoki:

- rezerwacje zalogowanego użytkownika (*indexuser.phtml*),
- wszystkie rezerwacje - widok administratorski (*indexadmin.phtml*),
- rezerwacje złożone (*indexr.phtml*),
- zaakceptowane rezerwacje (*indexa.phtml*),
- archiwalne zamówienia (*indexv.phtml*),
- dodanie nowego zamówienia (*add.phtml*),
- edycja zamówienia (*edit.phtml*).

```
<table class="table">
<tr>
    <th>Przedmiot</th>
    <th>Data złożenia zamówienia</th>
    <th>Status</th>
    <th>Informacje</th>
</tr>
<?php foreach ($zamowienias as $zamowienia) : ?>
<tr>
    <td><?php echo $this->escapeHtml($zamowienia->id_przedmiotu);?></td>
    <td><?php echo $this->escapeHtml($zamowienia->data);?></td>
    <td><?php echo $this->escapeHtml($zamowienia->status);?></td>
    <td><?php echo $this->escapeHtml($zamowienia->informacje);?></td>
```

Rysunek 9: Fragment pliku 'indexuser.phtml'



Twoje zamówienia			
Przedmiot	Data złożenia zamówienia	Status	Informacje
BG005	2016-01-07	Zaakceptowane	

Rysunek 10: Widok 'indexuser.phtml' w przeglądarce

Dla widoku *add.phtml* oraz *edit.phtml* należy wykorzystać wcześniej utworzoną formę w pliku *ZamowieniaForm.php*. Pozwoli ona na wyświetlenie danych kolumn z możliwością ich edycji (przy pomocy wcześniej zadeklarowanych typów).

```

</tr>
<?php
echo $this->form()->openTag($form); ?>
<p><?php echo $this->formHidden($form->get('id_zamowienia')); ?></p>
<tr>
    <td><p><?php echo 'Użytkownik: '; ?></p></td>
    <td><p><?php echo $this->formRow($form->get('id_uzytkownika')); ?></p></td>
</tr>
<tr>
    <td><p><?php echo 'Przedmiot: '; ?></p></td>

```

Rysunek 11: Fragment pliku 'edit.phtml'

Użytkownik	Przedmiot	Data złożenia zamówienia	Status	Informacje	Akcja
DDD	BG002	2016-01-07	Zaakceptowane		Edytuj
OS_Cypek	BG005	2016-01-07	Zaakceptowane		Edytuj

Rysunek 12: Widok administratorski 'indexa.phtml' w przeglądarce

## g. Wykonanie czynności autoryzacyjnych

Czynnością, którą należy zrobić na zakończenie tworzenia modułu, jest określenie dostępu każdego typu użytkownika (niezalogowanego, zalogowanego oraz administratora) do wszystkich widoków i akcji naszego modułu. W przypadku frameworku Zend2 zastosowano pomocnicze moduły ZFcUser oraz BjyAuthorise.

```

array('controller' => 'Składzik\Controller\Składzik', 'action' => 'books', 'roles' => array('guest', 'user')),
array('controller' => 'Składzik\Controller\Składzik', 'action' => 'stuff', 'roles' => array('guest', 'user')),
array('controller' => 'Uzytkownicy\Controller\Uzytkownicy', 'action' => 'index', 'roles' => array('admin')),
array('controller' => 'Uzytkownicy\Controller\Uzytkownicy', 'action' => 'edit', 'roles' => array('admin')),
array('controller' => 'Uzytkownicy\Controller\Uzytkownicy', 'action' => 'add', 'roles' => array('admin')),
array('controller' => 'Zamowienia\Controller\Zamowienia', 'action' => 'add', 'roles' => array('admin')),
array('controller' => 'Zamowienia\Controller\Zamowienia', 'action' => 'edit', 'roles' => array('admin')),
array('controller' => 'Zamowienia\Controller\Zamowienia', 'action' => 'index', 'roles' => array('admin')),
array('controller' => 'Zamowienia\Controller\Zamowienia', 'action' => 'indexuser', 'roles' => array('user')),

```

Rysunek 13: Fragment pliku 'bjyauthorize.global.php'

## h. Podłączenie modułu do aplikacji

Ostatnią czynnością, jaką należy wykonać, jest dołączenie naszego modułu poprzez dopisanie go do tablicy modułów w pliku *application.config.php* w folderze *config/autoload*.



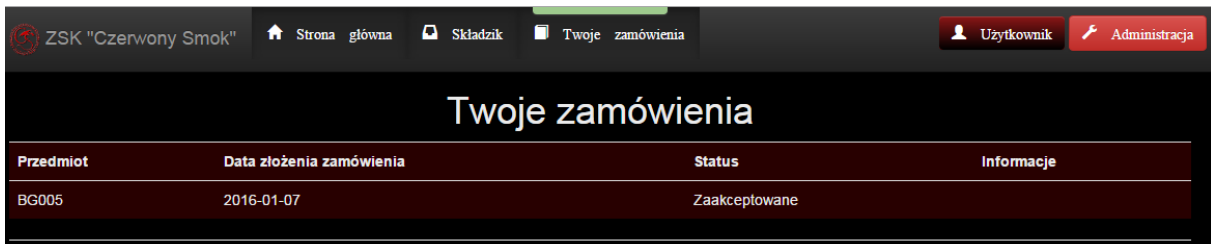
```

return array(
    // Używane przez aplikację moduły.
    'modules' => array(
        'Application',
        'ZfcBase',
        'ZfcUser',
        'ZfcAdmin',
        'BjyAuthorize',
        'Skladzik',
        'Zamowienia',
        'Uzytkownicy',
        // 'ZfcUserAuthentication',
        // 'Games',
        // 'Books',
    ),
);

```


Rysunek 14: Fragment pliku 'application.config.php' - deklaracja modułów

## 6. Prezentacja działania modułu



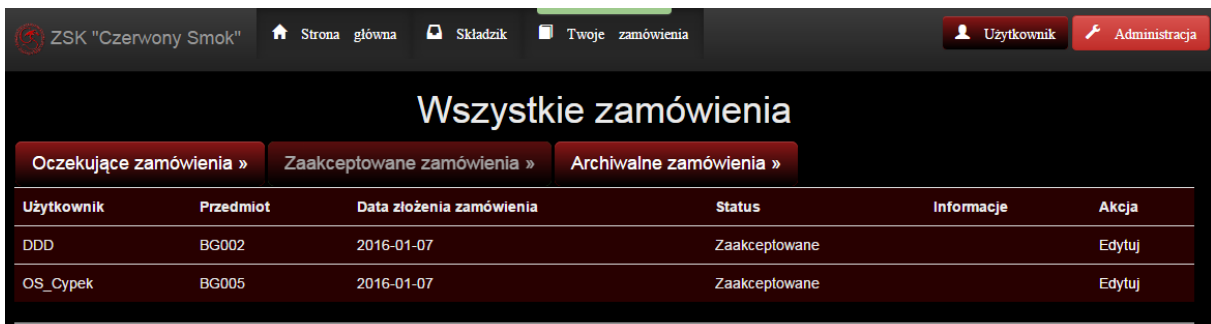
ZSK "Czerwony Smok"	Strona główna	Skladzik	Twoje zamówienia	Użytkownik	Administracja
Twoje zamówienia					
Przedmiot	Data złożenia zamówienia	Status	Informacje		
BG005	2016-01-07	Zaakceptowane			

Rysunek 15: Widok zamówień danego użytkownika (indexuser.phtml)



ZSK "Czerwony Smok"	Strona główna	Skladzik	Twoje zamówienia	Użytkownik	Administracja
Wszystkie zamówienia					
Oczekujące zamówienia »		Zaakceptowane zamówienia »		Archiwalne zamówienia »	
Użytkownik	Przedmiot	Data złożenia zamówienia	Status	Informacje	Akcja
DDD	BG002	2016-01-07	Oczekujące		Edytuj

Rysunek 16: Administratorski widok oczekujących zamówień (indexr.phtml)



ZSK "Czerwony Smok"	Strona główna	Skladzik	Twoje zamówienia	Użytkownik	Administracja
Wszystkie zamówienia					
Oczekujące zamówienia »		Zaakceptowane zamówienia »		Archiwalne zamówienia »	
Użytkownik	Przedmiot	Data złożenia zamówienia	Status	Informacje	Akcja
DDD	BG002	2016-01-07	Zaakceptowane		Edytuj
OS_Cypek	BG005	2016-01-07	Zaakceptowane		Edytuj

Rysunek 17: Administratorski widok zaakceptowanych zamówień (indexa.phtml)

ZSK "Czerwony Smok"	Strona główna	Składzik	Twoje zamówienia	Użytkownik	Administracja
Wszystkie zamówienia					
Oczekujące zamówienia »		Zaakceptowane zamówienia »		Archiwalne zamówienia »	
Użytkownik	Przedmiot	Data złożenia zamówienia		Status	Informacje
DDD	BG002	2016-01-18		Archiwalne	

Rysunek 18: Administratorski widok archiwalnych zamówień (indexv.phtml)

ZSK "Czerwony Smok"	Strona główna	Składzik	Twoje zamówienia	Użytkownik	Administracja
Dodaj nowe zamówienie					
Index	Wartość				
Użytkownik:	<input type="text"/>				
Przedmiot:	<input type="text"/>				
Status:	Oczekujące ▾				
Informacje:	<input type="text"/>				
<input type="button" value="Dodaj"/>					

Rysunek 19: Administratorski widok dodania nowego zamówienia (add.phtml)

ZSK "Czerwony Smok"	Strona główna	Składzik	Twoje zamówienia	Użytkownik	Administracja
Edytuj zamówienie					
Index	Wartość				
Użytkownik:	DDD <input type="text"/>				
Przedmiot:	BG002 <input type="text"/>				
Status:	Oczekujące ▾				
Informacje:	<input type="text"/>				
<input type="button" value="Zmień"/>					

Rysunek 20: Administratorski widok edycji zamówienia (edit.phtml)