



Aplikacje internetowe

Opis wybranej części wykonanej aplikacji

Imię i nazwisko: Konrad Niżnik

Numer albumu: 127905

Grupa: F1A-DU-L2

1. Ogólnie o aplikacji

Nazwa aplikacji:

SH Books

(SH to skrót od second hand)

Opis aplikacji:

Portal ogłoszeniowy do wymiany i sprzedaży używanych książek.

Frameworki:

Symfony, Bootstrap

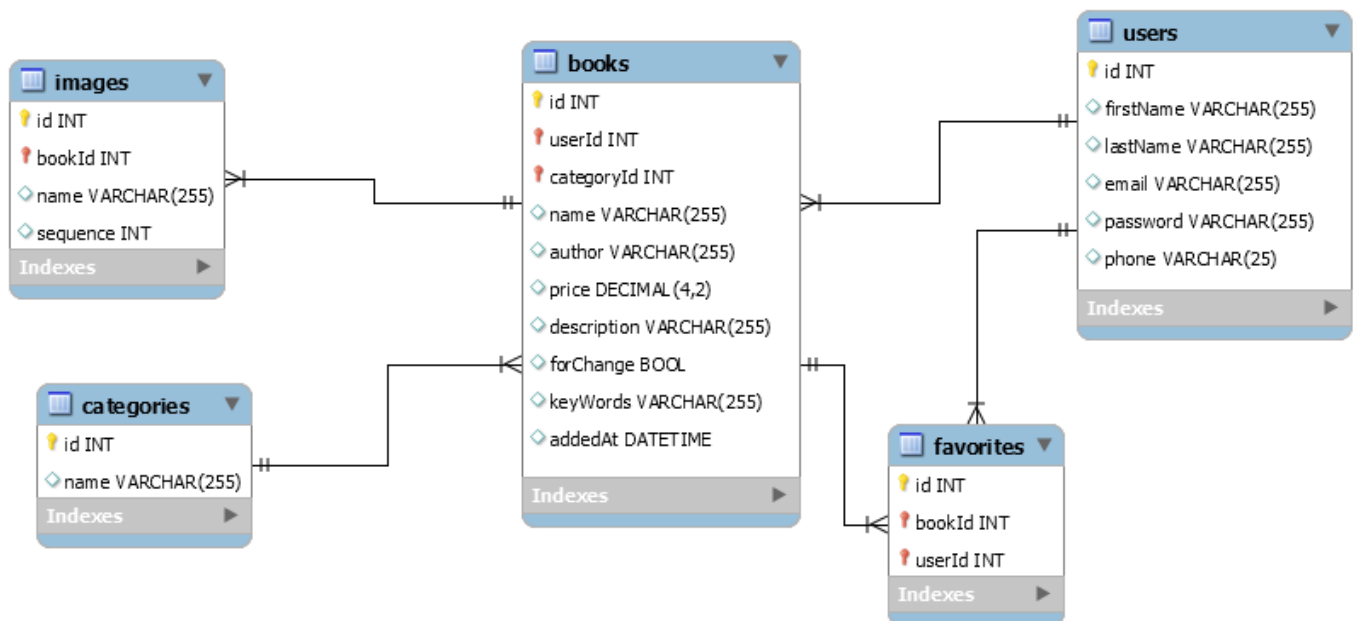
Technologie:

PHP, JS, HTML, CSS, Twig

Funkcjonalności:

- dodawanie ogłoszeń (wraz ze zdjęciami),
- przeglądanie ogłoszeń,
- wyszukiwanie ogłoszeń (po nazwie kategorii, tytule książki, autorze, bądź słowie kluczowym),
- logowanie,
- wylogowywanie.

Diagram ERD:



2. Dodawanie nowych książek

Dodawanie nowych książek jest możliwe po zalogowaniu się użytkownika i następnym kliknięciu na „Sprzedaj” w prawym górnym rogu aplikacji. Ukazuje nam się wtedy formularz dodawania książki, przedstawiony na zdjęciu poniżej.

SH Books Kategorie ▾

Tytuł, autor lub słowo kluczowe 🔍 Damian Nowak ▾ \$ Sprzedaj

+ Add files... ⌂ Cancel upload 🗑 Delete

Tytuł książki
Tytuł książki

Opis
Tytuł

Kategoria
Wybierz kategorię ▾

☐ Wymienie

Autor
Autor

Słowa kluczowe
Słowa kluczowe

Cena
Cena

Wyślij




Zdjęcie 1: Formularz dodawania książek.




Za pomocą przycisku „Add files...” możliwy jest wybór zdjęcia z dysku i jeśli jego format i rozmiar zgadza się ze zdefiniowanymi w kodzie wartościami (jest w formacie png, jpg, jpeg lub gif oraz nie jest większe niż 1100MB), zdjęcie zostanie wysłane na serwer, a jego podgląd zostanie dla użytkownika wyświetlony w formularzu, co widać na zdjęciu poniżej. Gdyby format lub rozmiar się nie zgadzał wyświetlony zostałby stosowny komunikat. Również gdyby spróbowano wgrać czwarte zdjęcie, system poinformowałby, że możliwe jest wgranie jedynie trzech zdjęć (tak ustawiono w kodzie).

SH Books Kategorie ▾

Tytuł, autor lub słowo kluczowe 🔍 Damian Nowak ▾ \$ Sprzedaj

+ Add files... ⌂ Cancel upload 🗑 Delete

Tytuł książki
Tytuł książki

Opis
Tytuł

Kategoria
Wybierz kategorię ▾

d
dla dzieci
dla młodzieży
Kuchnia i diety

Autor
Autor

Słowa kluczowe
Słowa kluczowe

Cena
Cena

Wyślij

Zdjęcie 2: Formularz dodawania książek (z dodanymi zdjęciami).

Kliknięcie w przycisk „Delete” obok przestanego zdjęcia, powoduje jego usunięcie z serwera i tym samym jego podglądu. Możliwe jest również usunięcie kilku zdjęć naraz poprzez zaznaczenie wybranych zdjęć checkboxem i kliknięcie przycisku „Delete”, znajdującego się powyżej wgranych zdjęć.

Na zdjęciu 2 widać również w jak wygodny sposób wybierana jest kategoria książki. Zwykły select zastąpiono selectem z biblioteki Chosen – jQuery plugin. Wystarczy zacząć wpisywać pożądaną kategorię, a plugin wyświetli tylko te opcje, które pasują do wpisanego w nim tekstu, następnie należy na nią kliknąć, a zostanie zaznaczona.

Cały formularz przed wysłaniem na serwer, podlega procesowi walidacji. Jak widać na poniższym zdjęciu, po wpisaniu w polu „Cena” na przykład wartość „20,999”, wyświetlony zostaje komunikat, że cena musi być w formacie „100” lub „100,00”.

Cena

- Cena musi być w formacie "100" lub "100,00."

Zdjęcie 3: Przykład błędu walidacji formularza.

Wykonanie działającego formularza dodawania książek (wraz ze zdjęciami), było dla mnie jedną z trudniejszych do wykonania części całej aplikacji. Najtrudniejsze w tym formularzu było wysyłanie zdjęć na serwer, bez przeładowywania strony z jednoczesnym wyświetlaniem ich miniatur. W wykonaniu tego zadania bardzo pomocny okazał się bezpłatny plugin: jQuery File Upload. Po jego zaimplementowaniu do mojego projektu, uzyskałem bardzo prosty w obsłudze dla użytkownika aplikacji widok dodawania kolejnych zdjęć do ogłoszenia. Największym wyzwaniem dla mnie, było „okodzenie” tego pluginu tak, aby on rzeczywiście działał (przesyłał zdjęcia na serwer), bo sama biblioteka ułatwiła mi jedynie stworzenie widoku.

Do poprawnego działania formularza potrzebne było stworzenie, między innymi:

- Metody kontrolera (addEditBookAction, która znajduje się w: src/AppBundle/Controller/BookActionsController.php) – zarządzającej całością działania formularza.

```
/**
 * Renders the book add or edit form.
 *
 * @Route("/add-edit-book/{bookId}", name="addEditBook", requirements={"bookId": "\d+"})
 */
public function addEditBookAction(Request $request, $bookId = NULL) {
    //Pobranie obiektu zalogowanego użytkownika
    $user = $this->getUser();

    //Jeżeli użytkownik nie jest zalogowany, to system przekierowuje go do formularza
    logowania
    if (!$user) {
```

```

        return $this->redirectToRoute('login');
    }

    //Pobranie obiektu Entity Manager - potrzebnego do wywoływania metod z plików
    znajdujących się w src/Appbundle/Repository (metody, te wykonują operacje na bazie danych)

    $em = $this->getDoctrine()->getManager();

    //Jeżeli nie podano id książki w parametrze tej metody, to zostaje stworzone nowe id
    książki i metoda wywoływana jest ponownie z nowym id.

    if (!$bookId) {

        return $this->redirectToRoute('addEditBook', array('bookId' => $em-
        >getRepository('AppBundle:books')->makeBook($user)));
    }

    //Inicjalizuje format daty oraz format liczb.

    ConstantsHelper::initialize($this->get('twig')->getExtension('core'));

    //Tworzy obiekt encji potrzebny do wygenerowania formularza wyszukiwania. Formularz
    wyszukiwania zawiera tylko jedno pole - pole wyszukiwania i w innych metodach jest ono
    wypełniane wpisanym wcześniej tekstem.

    $search = FormHelper::createSearchEntity();

    //Tworzy formularz wyszukiwania.

    $form = $this->createForm(\AppBundle\Form\SearchForm::class, $search);

    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {

        //Gdy formularz wyszukiwania książki został wysłany (bez błędów walidacji), system
        wyświetla wyniki wyszukiwania (wypełniając pole wyszukiwania wpisanym tekstem, zapisanym w
        zmiennej $searchArray).

        $searchArray = FormHelper::prepareSearchArray($form->getData());

        return $this->redirectToRoute('books', $searchArray);
    }

    //Tworzy formularz dodawania książki, wypełniając, go wartościami zapisanymi w bazie
    danych - (przydaje się w edycji ogłoszenia), jeśli w bazie danych nie ma informacji na
    temat danej książki (nie ma gdy książka jest dodawana), tworzony jest pusty formularz.

    $bookForm = $this->_createBookForm($em, $bookId);

    $bookForm->handleRequest($request);

    if ($bookForm->isSubmitted() && $bookForm->isValid()) {

        //Gdy formularz dodawania książki został wysłany (bez błędów walidacji), informacje
        wprowadzone w formularzu zostają zapisane w bazie danych

        $em->getRepository('AppBundle:books')->update($bookForm->getData(), $bookId,
        date(ConstantsHelper::$usaDateFormat));

        //Następuje przekierowanie do strony głównej.

        return $this->redirectToRoute('homepage');
    }

```

```
//Generuje widok ekranu z formularzem dodawania książki (w przyszłości plik
addEditBook.html.twig planuje wykorzystywać również do generowania widoku edycji książki -
stąd nazwa pliku)
```

```
return $this->render('page/addEditBook.html.twig', array(
    'form' => $form->createView(),
    'bookForm' => $bookForm->createView(),
    'bookId' => $bookId
));
}
```

- Klasy formularza (src/AppBundle/Form/BookForm.php) – do zdefiniowania pól formularza wraz z ich labelami, typami (np. text, number) oraz atrybutami (np. „class”, „placeholder”, „required”).

```
class BookForm extends AbstractType {
```

```
    /**
     * Pozwala na użycie (w tej klasie) zdefiniowanych w kontrolerze kategorii książek.
     */

    public function configureOptions(\Symfony\Component\OptionsResolver\OptionsResolver
$resolver) {
        parent::configureOptions($resolver);
        $resolver->setRequired(array('categories'));
    }

    /**
     * Zwraca gotowy do użycia formularz dodawania/edycji książek.
     */

    public function buildForm(FormBuilderInterface $builder, array $options) {
        $builder
            ->add('name', null, array(
                'label' => 'Tytuł książki',
                'label_attr' => array('class' => 'control-label'),
                'attr' => array('class' => 'form-control input-sm', 'placeholder' => 'Tytuł
książki'))))
            ->add('author', null, array(
                'label' => 'Autor',
                'label_attr' => array('class' => 'control-label'),
                'attr' => array('class' => 'form-control input-sm', 'placeholder' =>
'Autor'))))
    }
```

```

->add('description', TextareaType::class, array(
    'label' => 'Opis',
    'label_attr' => array('class' => 'control-label'),
    'required' => FALSE,
    'attr' => array('class' => 'form-control input-sm', 'rows' => 2,
'placeholder' => 'Tytuł'))
->add('forChange', CheckboxType::class, array(
    'label' => 'Wymienię',
    'required' => FALSE))
->add('keyWords', null, array(
    'label' => 'Słowa kluczowe',
    'label_attr' => array('class' => 'control-label'),
    'required' => FALSE,
    'attr' => array('class' => 'form-control input-sm', 'placeholder' => 'Słowa
kluczowe'))))
->add('category', ChoiceType::class, array(
    'label' => 'Kategoria',
    'choices' => $options['categories'],
    'required' => FALSE,
    'label_attr' => array('class' => 'control-label'),
    'attr' => array('class' => 'chosen-select', 'data-placeholder' => 'Wybierz
kategorię'))))
->add('price', null, array(
    'label' => 'Cena',
    'label_attr' => array('class' => 'control-label'),
    'attr' => array('class' => 'form-control input-sm', 'placeholder' =>
'Cena'))))
->add('save', SubmitType::class, array(
    'label' => 'Wyślij',
    'attr' => array('class' => 'btn btn-lg btn-success')));
}

}

```

- Klasy encji formularza (src/AppBundle/Entity/Book.php) – (wykorzystywanej przez klasę formularza) zawierającej wszystkie pola formularza wraz z ich regułami walidacji.

```
class Book {
```

```
/**
```

```
    * @Assert\Type("string")
    * @Assert\NotBlank()
    */
```

```
public $name;
```

```
/**
 * @Assert\Type("string")
 * @Assert\NotBlank()
 */
```

```
public $author;
```

```
/**
 * @Assert\Length(
 *     min = 5,
 *     max = 255)
 */
```

```
public $description;
```

```
public $forChange;
```

```
/**
 * @Assert\Type("string")
 */
```

```
public $keyWords;
```

```
/**
 * @Assert\NotBlank()
 */
```

```
public $category;
```

```
/**
 * @Assert\Type("string")
 * @Assert\NotBlank()
 */
```

```
public $price;
```



```

/**
 * @Assert\Callback
 */

public function validate(ExecutionContextInterface $context, $payload) {
    if (!preg_match('/^[0-9]+(\,[0-9]{2})?$/ ', $this->price)) {
        $context->buildViolation('Cena musi być w formacie "100" lub "100,00."')
            ->atPath('price')
            ->addViolation();
    }
}
}

```

- Metody dodające informacje z wypełnionego poprawnie formularza do bazy danych (znajdują się w `src/AppBundle/Repository/booksRepository.php`).

```

/*
 * Zwraca nowy (pusty) obiekt książki, który następnie będzie wypełniony danymi z formularza.
 * Do dodawania zdjęć (bez przeładowywania strony) potrzebny jest bookId, który pobierany jest
 * ze zwróconego obiektu.
 */

public function makeBook($user) {
    $books = new books();
    $books->setUsers($user);
    $em = $this->getEntityManager();
    $em->persist($books);
    $em->flush();
    return $books->getId();
}

/*
 * Dodaje/zmienia dane dotyczące ogłoszenia do bazy.
 */

public function update($data, $bookId, $today) {
    $em = $this->getEntityManager();
    $query = $em->createQuery(
        "UPDATE AppBundle:books b
        SET b.name = :name, b.author = :author, b.description = :description, b.forChange =
        :forChange,

```

```

        b.keyWords = :keyWords, b.price = :price, b.categories = :categoryId, b.addedAt =
:today

        WHERE b.id = :bookId"
    )->setParameters(array(
        'bookId' => $bookId,
        'name' => $data->name,
        'author' => $data->author,
        'description' => $data->description,
        'forChange' => $data->forChange,
        'keyWords' => $data->keyWords,
        'price' => $data->price,
        'categoryId' => $data->category,
        'today' => new \DateTime($today)
    ));
    $query->execute();
}

```

- Serwis (src/Appbundle/Services/Upload.php) zawierający metody do zapisu i usuwania zdjęć, tworzenia miniaturk zdjęć, pobierania wysłanych już zdjęć.
- Pliku generującego widok formularza (app/Resources/views/page/addEditBook.html.twig – rozbudowuje on plik: app/Resources/views/page/_fileUploadBase.html.twig, który z kolei rozbudowuje: app/Resources/views/base.html.twig).

3. Podsumowanie

Projekt polegał na wykonaniu portalu ogłoszeniowego do wymiany i sprzedaży używanych książek, jego założenia zostały zrealizowane pomyślnie. Dzięki wykonaniu tego projektu, poszerzyłem swoją wiedzę na temat frameworka PHP Symfony oraz nauczyłem się obsługi dwóch, bardzo użytecznych bibliotek: „jQuery File Upload” (do wgrywania plików na serwer, bez przeładowywania strony) oraz „Chosen” (zastępującego standardowy select, bardziej użytecznym).