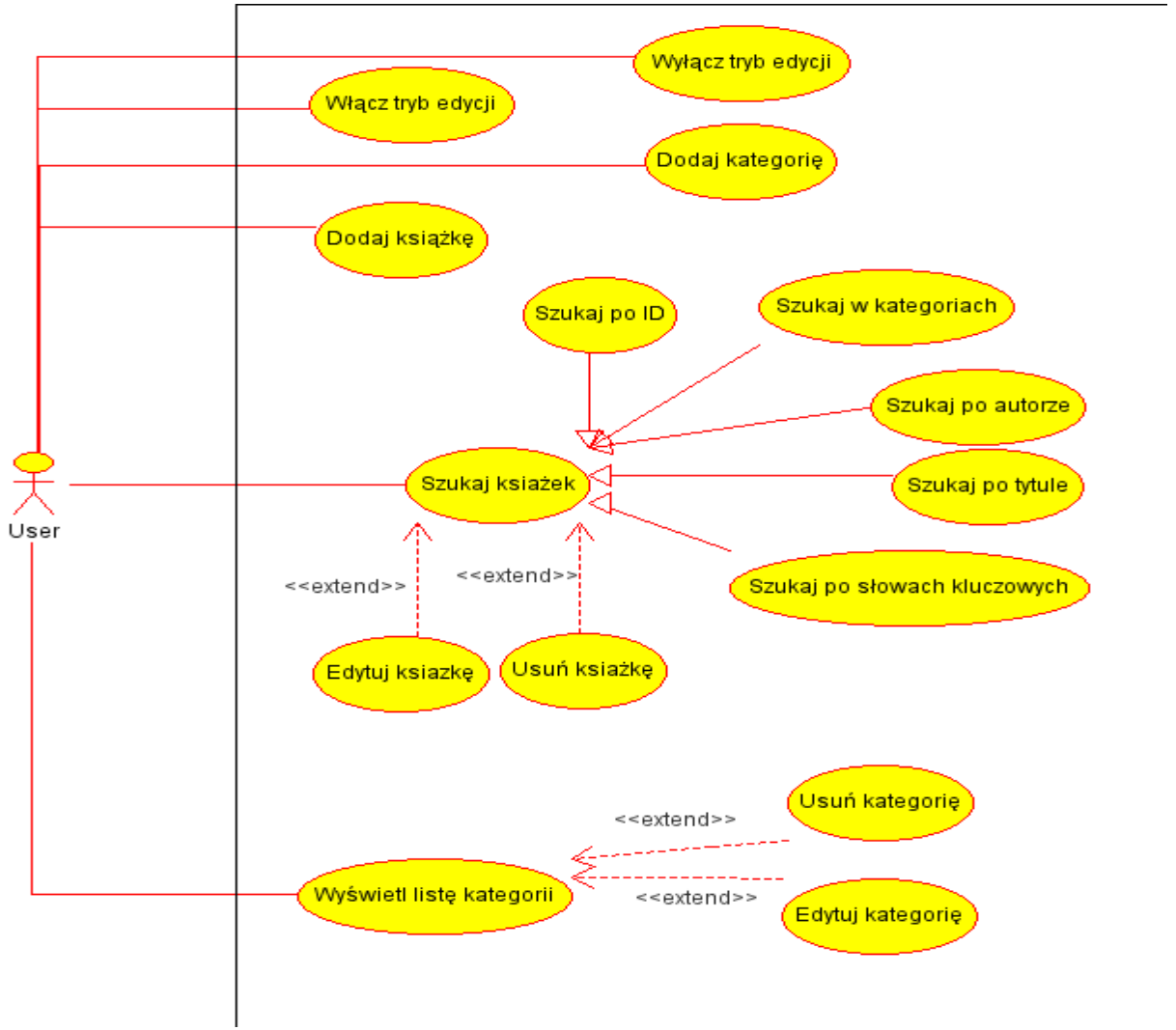
Podstawy programowania III

WYKŁAD 7

Jan Kazimirski

Projekt: „Katalog książek elektronicznych” c.d.

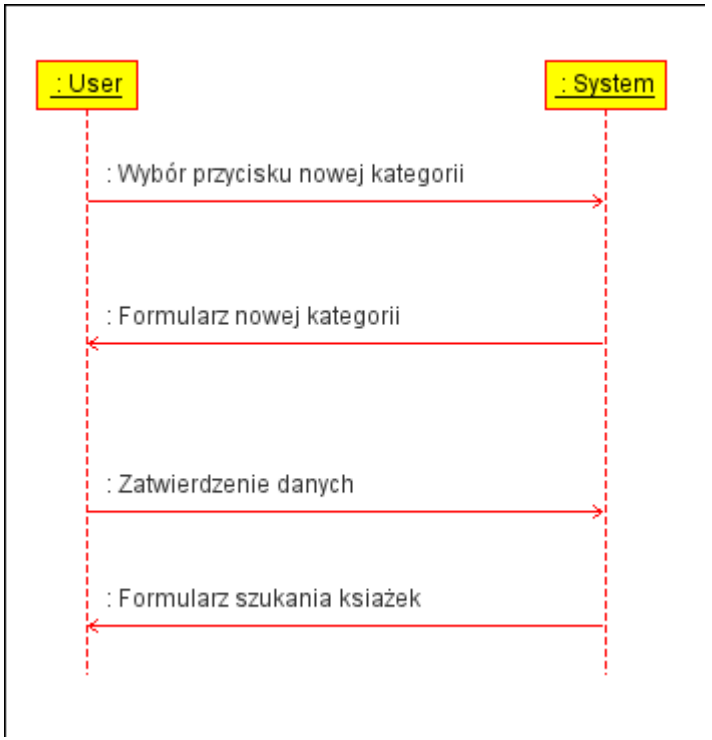
Diagram przypadków użycia



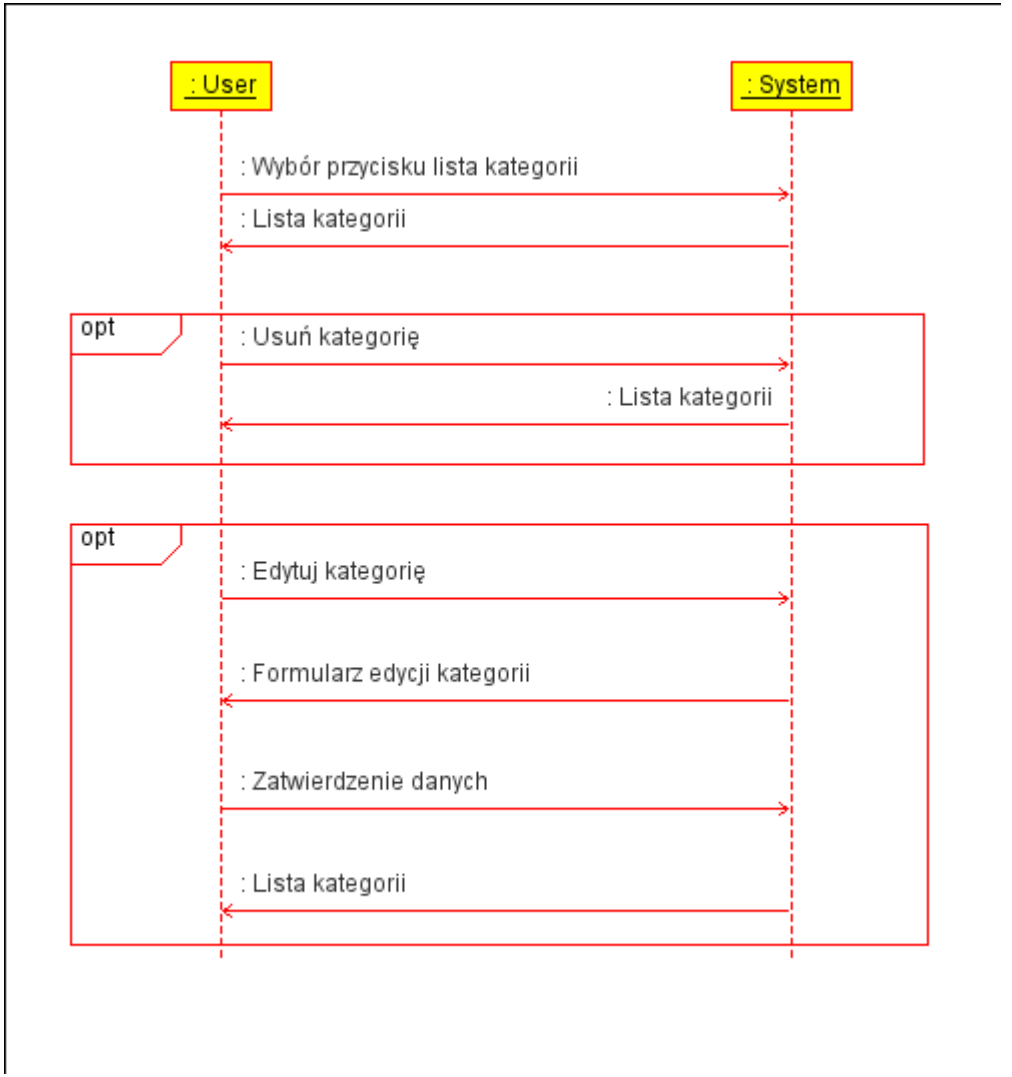
Iteracja 2

- Realizacja przypadków użycia
  - Dodaj kategorię
  - Wyświetl listę kategorii
  - Edytuj kategorię
  - Usuń kategorię

Obsługa kategorii

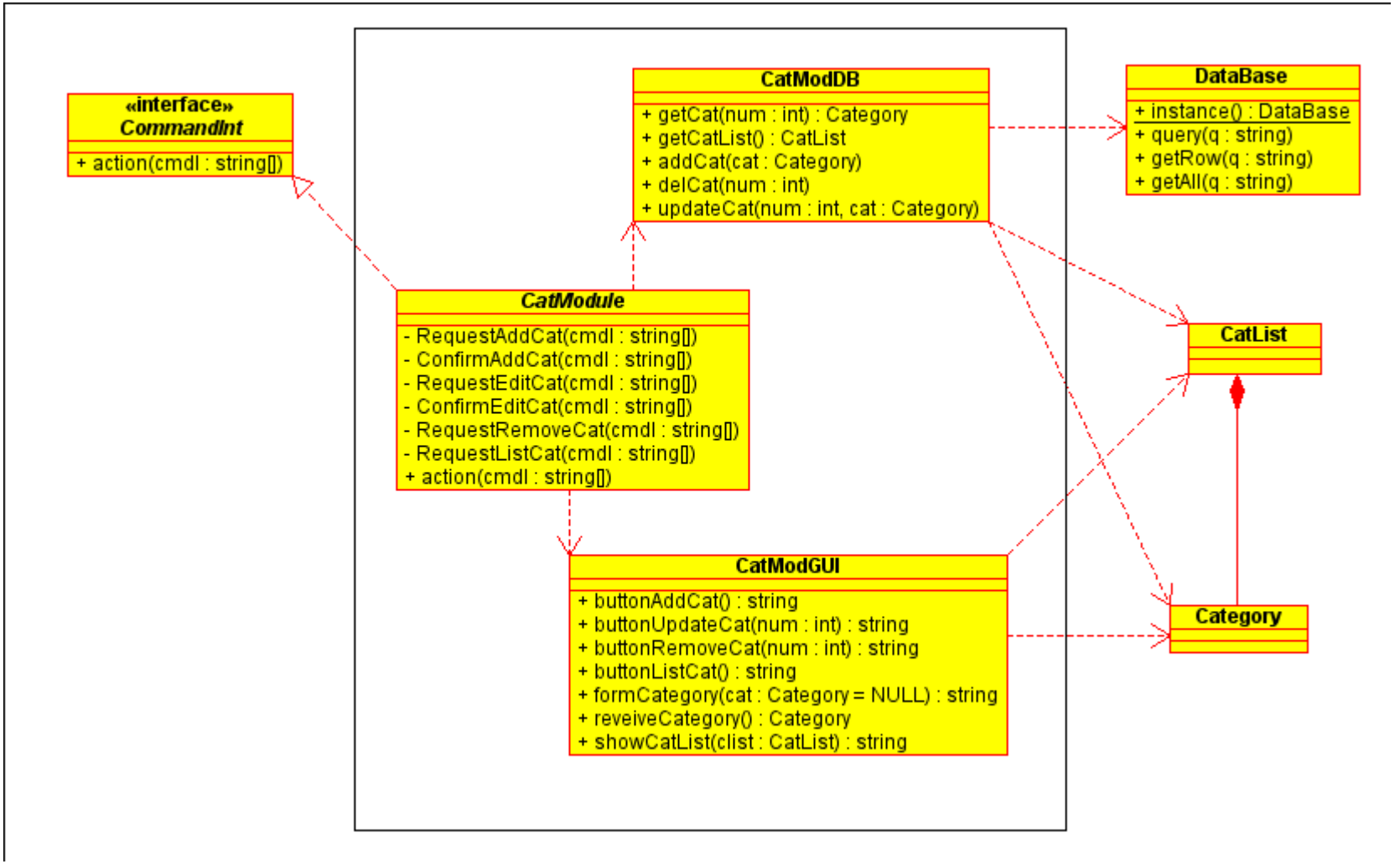


Dodanie kategorii



Lista kategorii, dodaj, usuń kategorię

Moduł CatModule - klasy



Iteracja 2 – stan obecny

- Zaimplementowane klasy:
  - DataBase
  - Category
  - CatList
  - CatModDB
- Pozostało do implementacji:
  - CatModGUI
  - CatModule

CatModGUI – szablon (usunięte metody button*)

```
class CatModGUI {

  public function formCategory() {}

  public function receiveCategory() {}

  public function showCatList() {}

};
```

UNIT TESTS - CatModGUI

testFormCategory

testReceiveCategory

testShowCatList

CatModGUI – implementacja (1)

```
public function formCategory(Category $cat=NULL) {
  if($cat==NULL) $cat = new Category("",-1);
  $str = "<h1>Add or update category</h1>";
  $str.= "<form action=".$_SERVER['PHP_SELF']." method=\"GET\">";
  $str.= "<input type=\"hidden\" name=\"cmd\" value=\"cr\">";
  $str.= "<input type=\"hidden\" name=\"cid\" value=\"".$cat->getId()."\">";
  $str.= "Category: <input type=\"text\" name=\"cname\" value=\"".$cat->getName()."\">";
  $str.= "<input type=\"submit\" value=\"Save\"></form>";
  return $str;
}
```

UNIT TESTS - CatModGUI

testFormCategory

testReceiveCategory

testShowCatList

CatModGUI – implementacja (2)

```
public function receiveCategory($data) {
  return new Category($data["cname"],$data["cid"]);
}
```

UNIT TESTS - CatModGUI

testFormCategory

testReceiveCategory

testShowCatList

CatModGUI – implementacja (3)

```
public function  showCatList(CatList $cl) {
  $edit = AccMode::isEditMode();
  $str = "<h1>Category list</h1>";
  $str.= "<table border>";
  $str.= "<tr><th>Nr</th><th>Name</th>";
  if($edit) $str.= "<th>Action</th>";
  $str.= "</tr>";
  $nr=0;
  foreach($cl as $cat) {
    $str.= "<tr><td>".++$nr."</td><td>".$cat->getName()."</td>";
    if($edit) {
      $id = $cat->getId();
      $str.= "<td><table><tr><td>".
            HtmlTools::formButton("Edit",array("cmd"=>"cu","cid"=>"$id")).
            "</td><td>".
            HtmlTools::formButton("Delete",array("cmd"=>"cd","cid"=>"$id")).
            "</td></tr></table></td>";
    };
    $str.= "</tr>";
  }
  $str.= "</table>";
  return $str;
}
```

UNIT TESTS - CatModGUI

testFormCategory

testReceiveCategory

testShowCatList

CatModule - szablon

```
class CatModule implements CommandInt {

  public function action($req) {}

  private function requestAddCat() {}

  private function receiveCat($data) {}

  private function requestEditCat($id) {}

  private function requestDeleteCat($id) {}

  private function requestListCat() {}
};
```

Fasada modułu obsługi kategorii – udostępnia funkcje modułu.

requestAddCat – użytkownik wybrał przycisk nowej kategorii.

receiveCat – użytkownik wypełnił i zatwierdził formularz nowej lub aktualizowanej kategorii

requestEditCat – użytkownik wybrał przycisk edycji kategorii

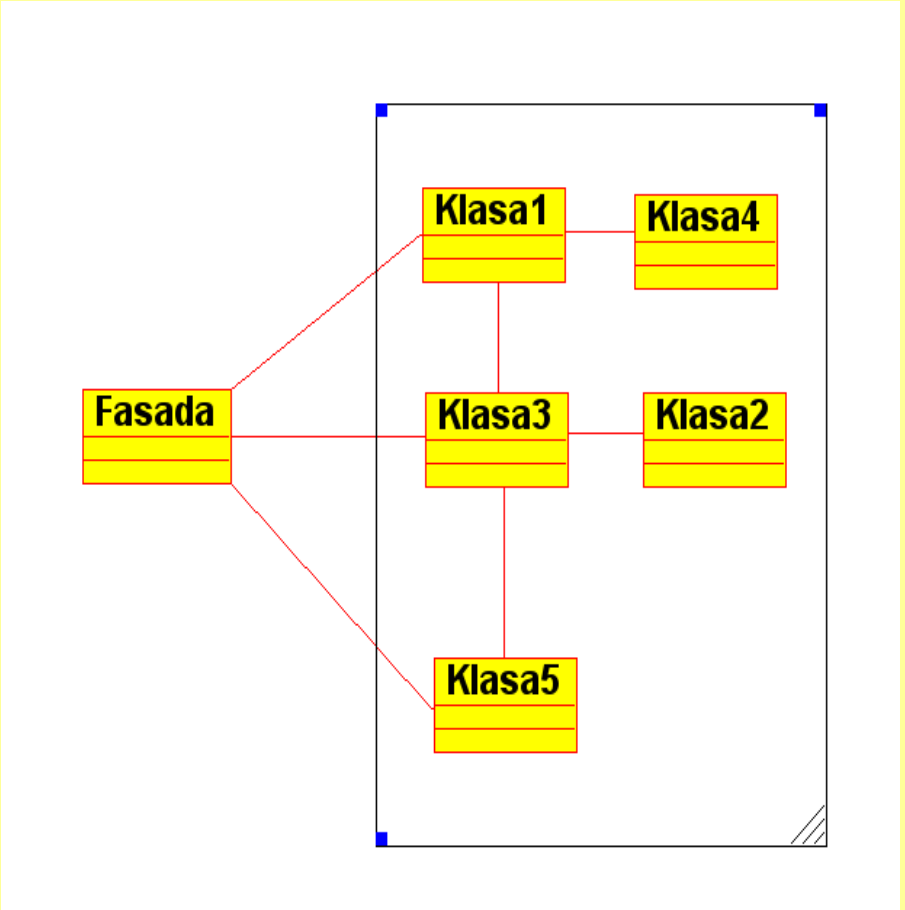
requestDeleteCat – użytkownik wybrał przycisk usunięcia kategorii

requestListCat – użytkownik wybrał przycisk wyświetlenia listy kategorii.

action – wybiera odpowiednie działanie na podstawie kodu polecenia przekazanego z przeglądarki ('cmd')

Zastosowanie Fasady

- Ukrywa złożony podsystem.
- Udostępnia prosty interfejs pozwalający na korzystanie z podsystemu bez znajomości jego szczegółów.
- Ukrywa implementację podsystemu – umożliwia łatwe zmiany.



CatModule – c.d

```
class CatModule implements CommandInt {

  public function action($req) {}

  private function requestAddCat() {}

  private function receiveCat($data) {}

  private function requestEditCat($id) {}

  private function requestDeleteCat($id) {}

  private function requestListCat() {}
};
```

UNIT TESTS - CatModGUI

- testActionRequestAddCat
- testActionReceiveNewCat
- testActionReceiveUpdatedCat
- testActionRequestEditCat
- testActionRequestDeleteCat
- testActionRequestListCat

CatModule - sterowanie

- Całą funkcjonalność modułu ukryta jest w metodzie **action** klasy CatModule.
- Tablica przekazywana jako argument do metody zawiera element 'cmd' którego wartość określa czynność.
- Kodowanie czynności (zawartość 'cmd'):
  - Pierwszy znak – określa moduł – '**c**'
  - Kolejne znaki określają czynności: '**a**' – dodaj nową kategorię (przycisk), '**r**' – odbierz dane kategorii (nowej lub edytowanej), '**e**' – edytuj kategorię , '**d**' – usuń kategorię , '**l**' – wyświetl listę kategorii.

CatModule – implementacja (1)

```
class CatModule implements CommandInt {
  public function action($req) {
    if($req["cmd"]=="ca") return self::requestAddCat();
    if($req["cmd"]=="cr") return self::receiveCat($req);
    if($req["cmd"]=="ce") return self::requestEditCat($req);
    if($req["cmd"]=="cd") return self::requestDeleteCat($req);
    if($req["cmd"]=="cl") return self::requestListCat();
  }

  private function requestAddCat() {
    return CatModGUI::formCategory();
  }
```

UNIT TESTS - CatModGUI

testActionRequestAddCat
testActionReceiveNewCat
testActionReceiveUpdatedCat
testActionRequestEditCat
testActionRequestDeleteCat
testActionRequestListCat

CatModule – implementacja (2)

```
private function receiveCat($req) {
  $cat = CatModGUI::receiveCategory($req);
  if($cat->getId()==-1) CatModDB::addCat($cat);
  else CatModDB::updateCat($cat);
  return self::requestListCat();
}

private function requestEditCat($req) {
  $cat = CatModDB::getCat($req["cid"]);
  return CatModGUI::formCategory($cat);
}
```

UNIT TESTS - CatModGUI

testActionRequestAddCat
testActionReceiveNewCat
testActionReceiveUpdatedCat
testActionRequestEditCat
testActionRequestDeleteCat
testActionRequestListCat

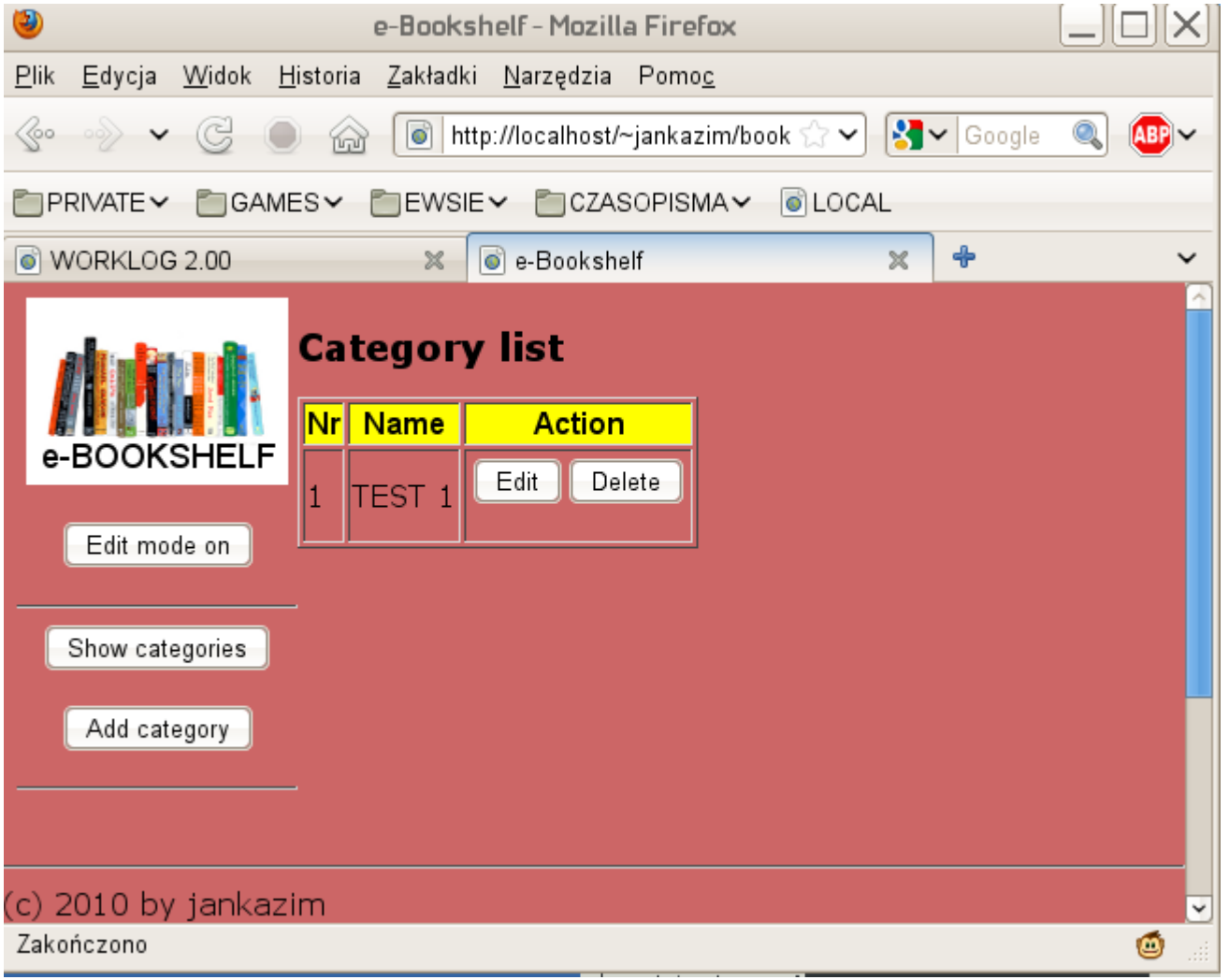
CatModule – implementacja (3)

```
  private function requestDeleteCat($req) {
    CatModDB::deleteCat($req["cid"]);
    return self::requestListCat();
  }

  private function requestListCat() {
    $cl = CatModDB::getCatList();
    return CatModGUI::showCatList($cl);
  }
};
```

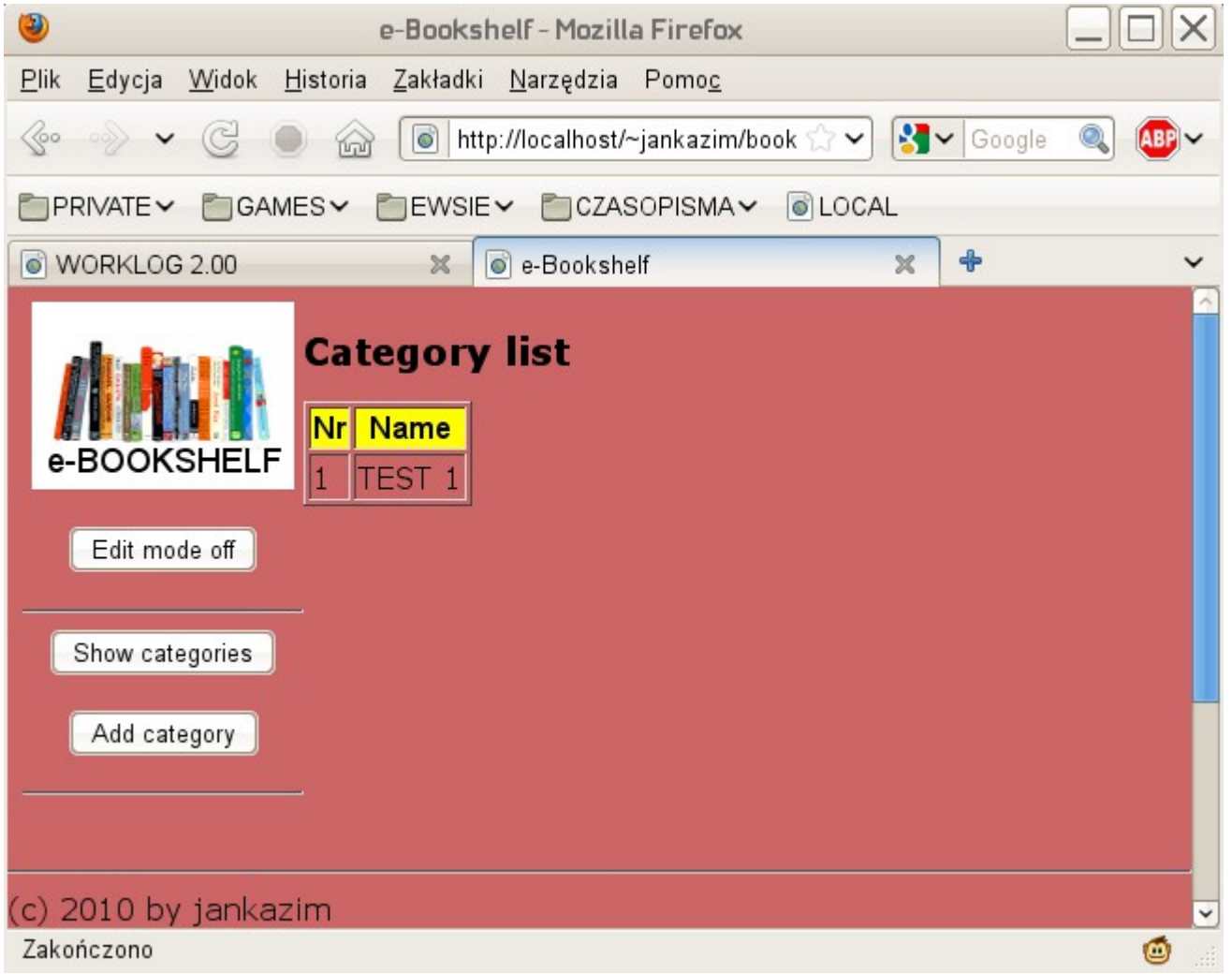
UNIT TESTS - CatModGUI

testActionRequestAddCat
testActionReceiveNewCat
testActionReceiveUpdatedCat
testActionRequestEditCat
testActionRequestDeleteCat
testActionRequestListCat

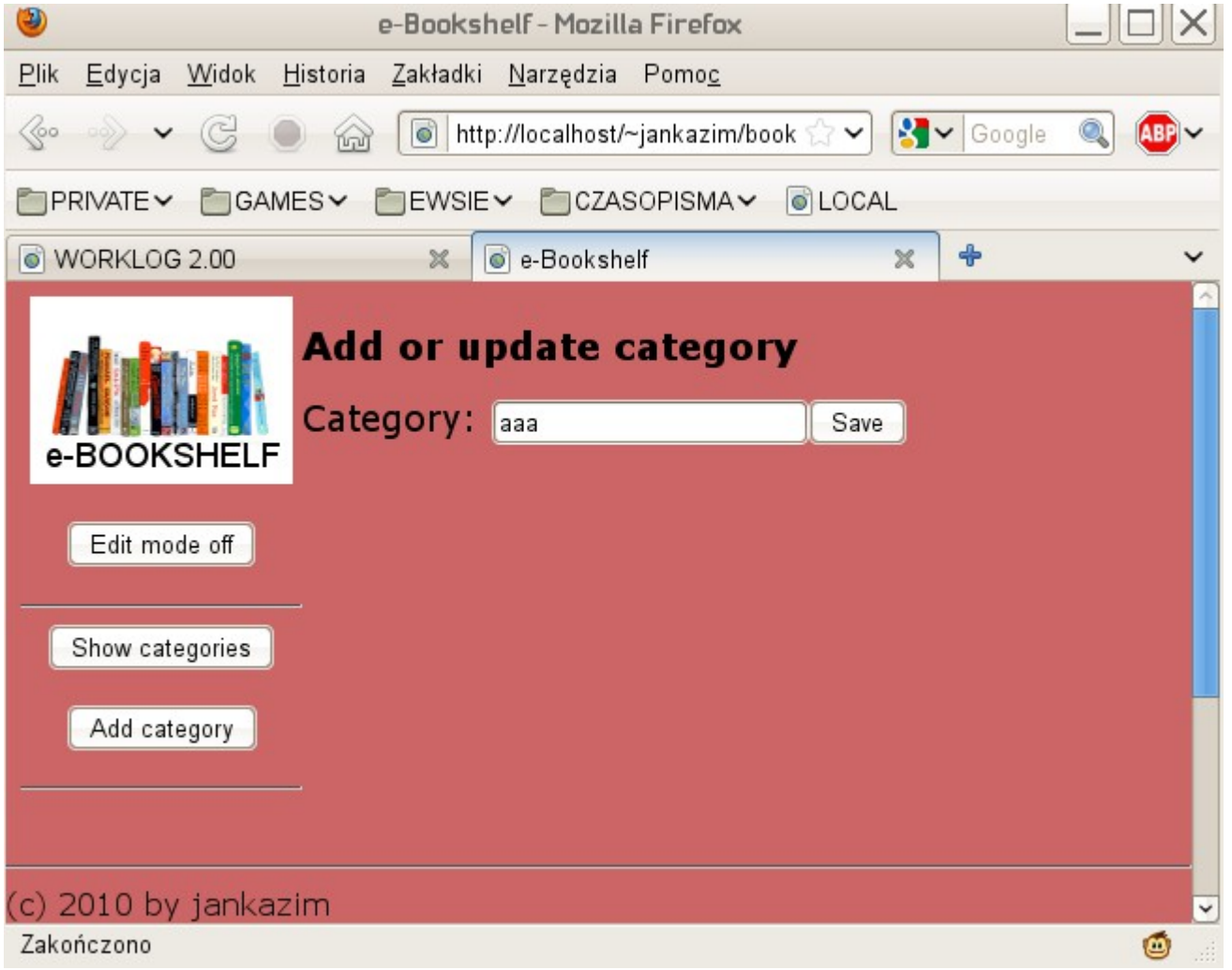
Moduł kategorii – działające GUI



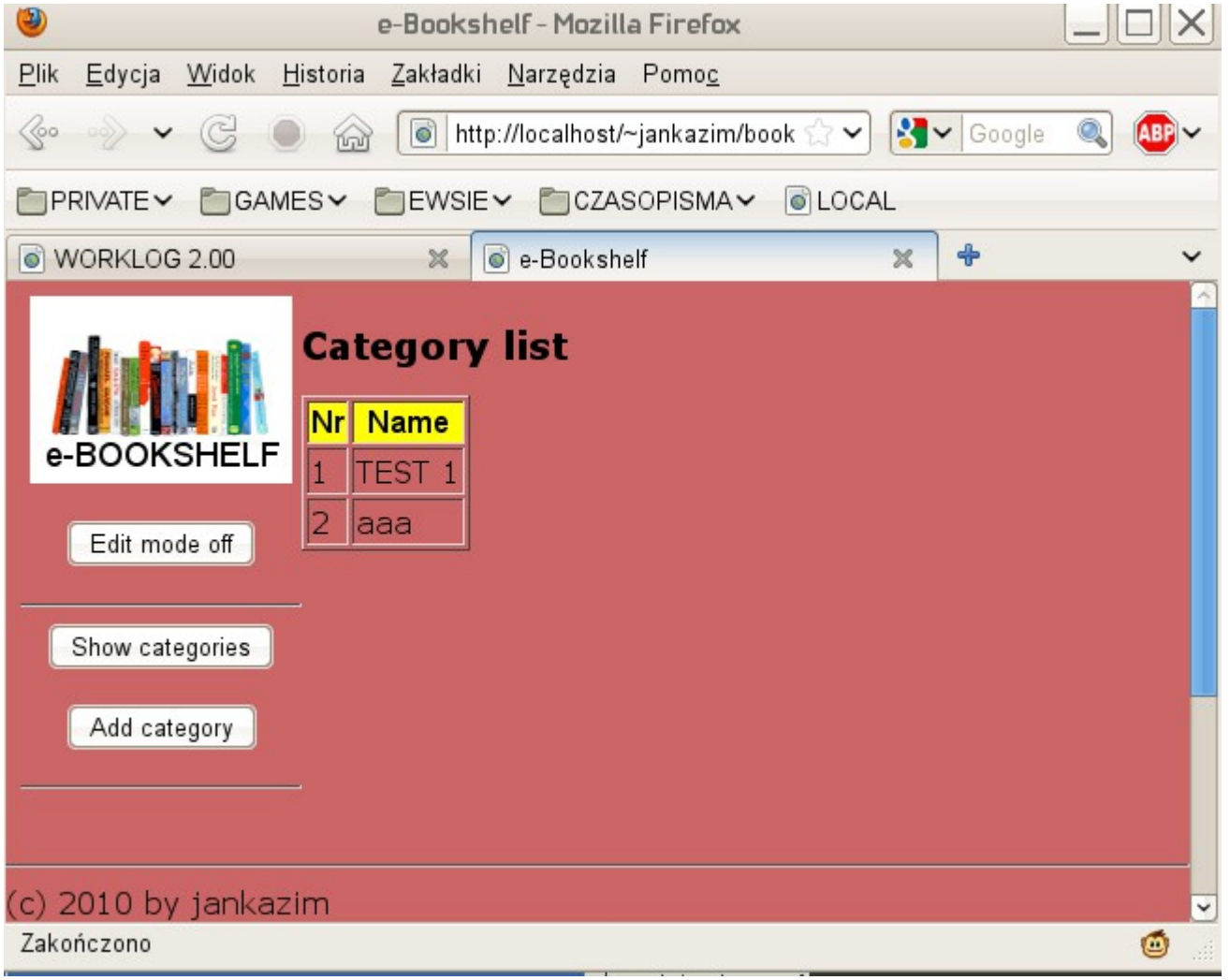
Moduł kategorii – działające GUI



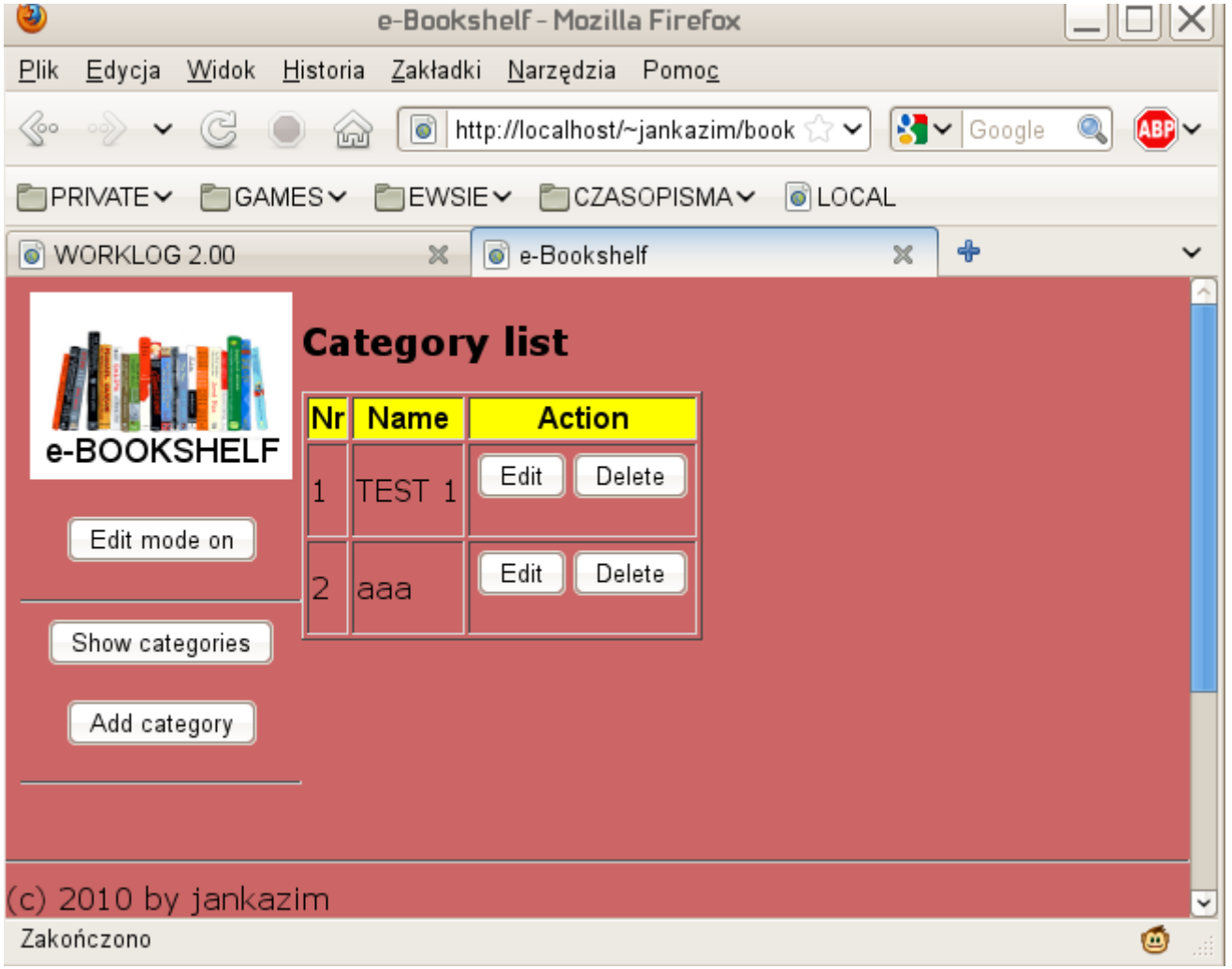
Moduł kategorii – działające GUI



Moduł kategorii – działające GUI



Moduł kategorii – działające GUI



Moduł kategorii - TODO

- Zaprojektować „profesjonalne” GUI
- Komunikaty o błędach w GUI (wychwytywanie wyjątków)
- Zmiana trybu edycji powoduje reset aktualnego polecenia (dotyczy bardziej AccMode niż CatModule).

Iteracja 3

- Realizacja przypadków użycia:
  - Dodaj książkę
  - Edytuj książkę
  - Usuń książkę
  - Częściowa realizacja przypadku Szukaj książki (wyświetlenie listy książek bez możliwości ustalania kryteriów wyszukiwania).

Diagram sekwencji

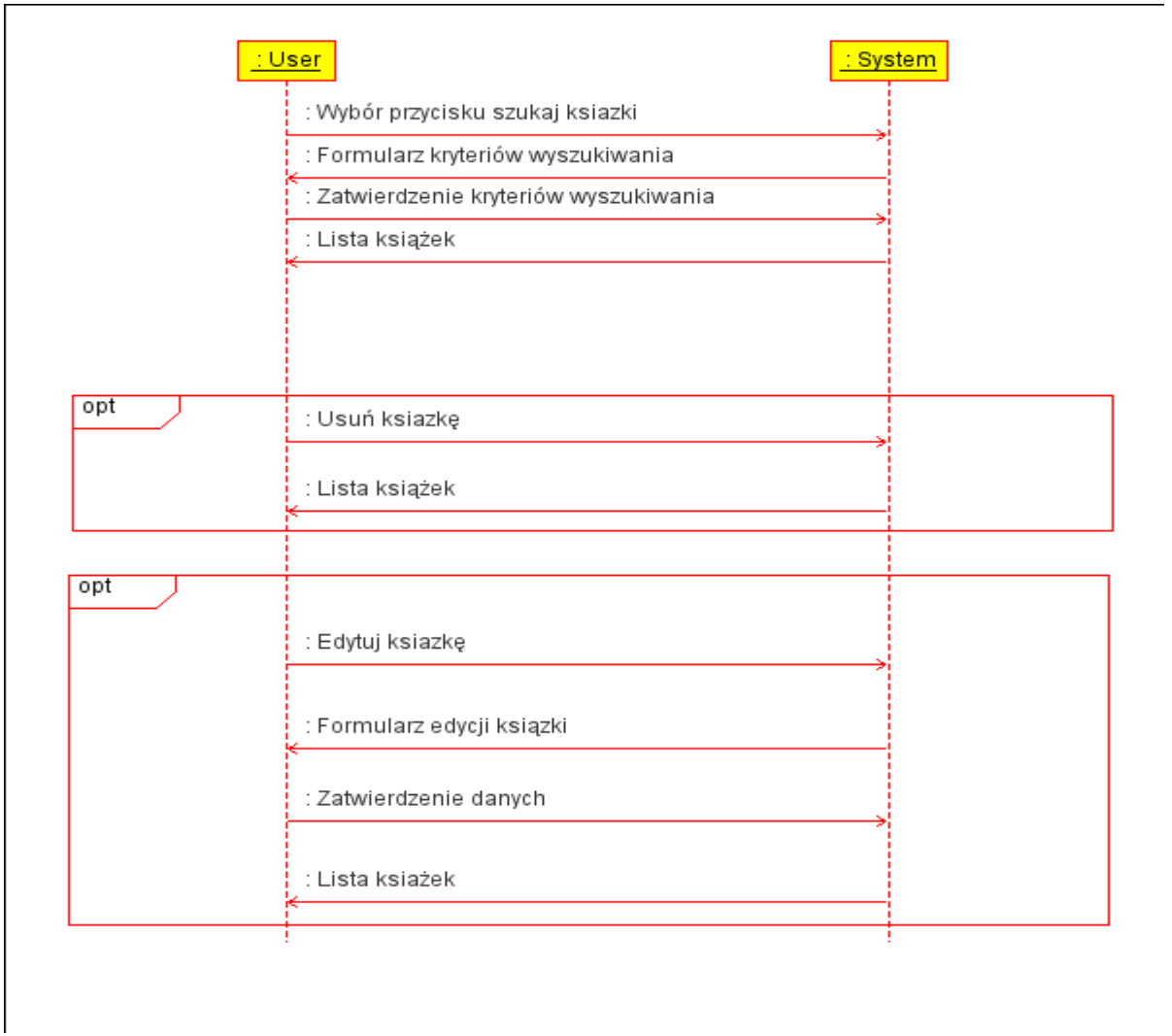
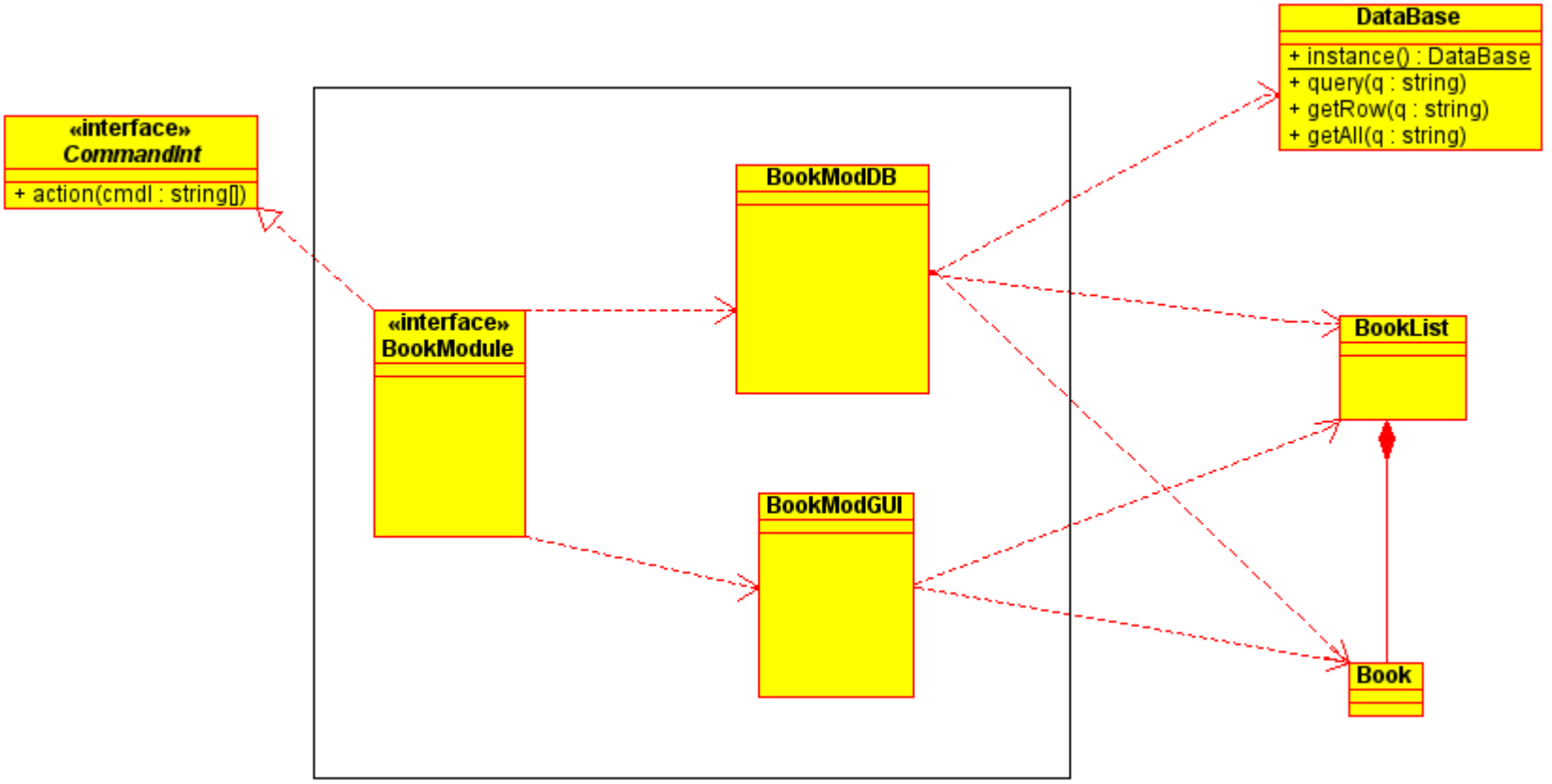


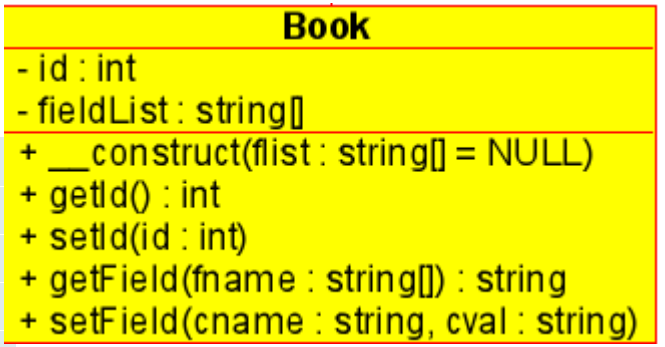
Diagram klas - szkielet



BookModule - szkielet

```php
<?php

require_once("DataBase.php");
require_once("CommandInt.php");
require_once("AccMode.php");

class Book {};

class BookList {};

class BookModDB {};

class BookModGUI {};

class BookModule implements CommandInt {
  public function action($req) {}
};

?>
```

Klasa Book




```php
class Book {
  private $bid;
  private $fieldList;

  public function __construct($t = NULL) {}
  public function getId() {}
  public function setId($id) {}
  public function getField($fname) {}
  public function setField($fname,$fval) {}
};
```

UNIT TESTS - Book

testGetSetId
testGetSetField
testConstrEmpty
testConstrInit

Klasa Book - implementacja

```
class Book {
  private $bid;
  private $fieldList;

  public function __construct($t = NULL) {}

  public function getId() {
    return $this->bid;
  }

  public function setId($id) {
    $this->bid = $id;
    $this->setField("id_b",$id);
  }

  public function getField($fname) {}
  public function setField($fname,$fval) {}

};
```

UNIT TESTS - Book

testGetSetId
testGetSetField
testConstrEmpty
testConstrInit

Klasa Book – implementacja c.d.

```
class Book {
  private $bid;
  private $fieldList;

  public function __construct($t = NULL) {
  }

  // getId,setId

  public function getField($fname) {
    return $this->fieldList[$fname];
  }

  public function setField($fname,$fval) {
    $this->fieldList[$fname]=$fval;
  }

};
```

UNIT TESTS - Book

testGetSetId
testGetSetField
testConstrEmpty
testConstrInit

Klasa Book – implementacja c.d.

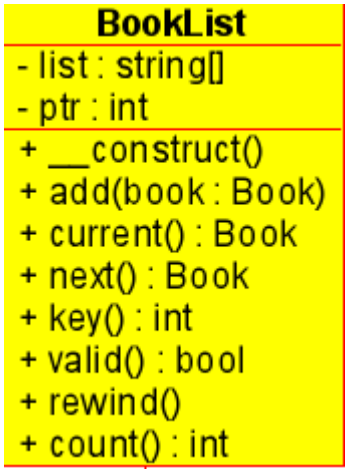
```
class Book {
  private $bid;
  private $fieldList;

  public function __construct($t = NULL) {
    $this->bid=-1;
    if($t!=NULL) {
      if(isset($t["id_b"])) $this->bid=$t["id_b"];
      $this->fieldList = $t;
    }
    if(!isset($this->fieldList["id_b"]))
        $this->setField("id_b",-1);
  }

  // getId,setId,getField,setfield

};
```

UNIT TESTS - Book

testGetSetId
testGetSetField
testConstrEmpty
testConstrInit

Klasa BookList

```
class BookList implements Iterator {
  private $list;
  public function __construct() {}
  public function add(Book $b) {}
  public function current() {}
  public function next() {}
  public function key() {}
  public function valid() {}
  public function rewind() {}
  public function count() {}
};
...
```



UNIT TESTS - BookList

testConstructor

testAddBook

testIterator

Kopiowanie kodu

- Klasa **BookList** jest prawie dokładną kopią klasy **CatList** z poprzedniej iteracji.
- Reguły OOP przestrzegają przed tworzeniem wielu kopii tego samego kodu.
- Kopiowanie kodu powoduje trudności w utrzymaniu (testowanie, poprawianie, modyfikowanie) wielu takich samych lub bardzo podobnych fragmentów kodu.

Kopiowanie kodu c.d.

- **DRY** – **D**on't **R**epeat **Y**ourself
  - Reguła zalecająca separowanie powtarzającego się kodu (np. osobne funkcje) i odwoływanie się do niego w razie potrzeby
- **ROT** – **R**ule **o**f **T**hree
  - Wprowadzona przez Martina Fowlera reguła programowania dopuszczająca jedno kopiowanie kodu, ale zalecająca wyizolowanie takiego kodu jeżeli liczba kopiowań jest większa.

CatList i BookList

- Zgodnie z zasadą oddzielnego tworzenia i refaktoryzowania kodu pozostawimy nasze implementacje bez zmiany.
- Mając zaimplementowany kod i gotowy zestaw testów dla obu klas należałoby się jednak zastanowić nad stworzeniem bardziej ogólnej klasy (np. List).
- Wysiłek związany z refaktoryzacją może się **bardzo** opłacić, bo uogólniona klasa List może być użyteczna w kolejnych projektach (obecne klasy wymagają drobnych przeróbek).

BookList - implementacja

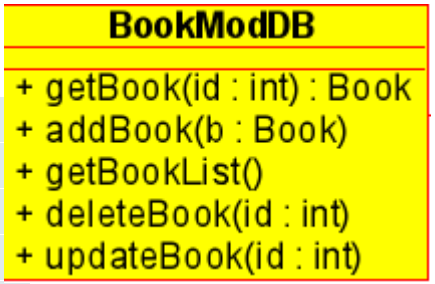
```
class BookList implements Iterator{
  private $list;
  private $ptr;

  public function __construct() {
    $this->list = array();
    $this->ptr = 0;
  }

  public function add(Book $b) {
    $this->list[] = $b;
  }

  public function current() {
    return $this->list[$this->ptr];
  }

  public function next() {
    $this->ptr++;
  }

  public function key() {
    return $this->list[$this->ptr]->getId();
  }

  public function valid() {
    return isset($this->list[$this->ptr]);
  }

  public function rewind() {
    $this->ptr = 0;
  }

  public function count() {
    return count($this->list);
  }

};
```

UNIT TESTS - BookList

testConstructor
testAddBook
testIterator

BookModDB

```
class BookModDB {

  public function getBook($id) {}

  public function addBook(Book $b) {}

  public function getBookList() {}

  public function deleteBook($id) {}

  public function updateBook(Book $b) {}
};
```



UNIT TESTS - BookModDB

- testDbGetBook
- testDbGetBookList
- testDbAddBook
- testDbDeleteBook
- testDbUpdateBook
- testDbGetBookInvalidId
- testDbDeleteBookInvalidId

BookModDB - implementacja

```
public function getBook($id) {
    $db = DataBase::instance();
    $res=$db->getRow("select * from bookshelf_booklist where id_b=$id;");
    if($res==FALSE) throw new Exception("InvalidBookID");
    return new Book($res);
  }

  public function addBook(Book $b) {
    $author=$b->getField("author");
    $title=$b->getField("title");
    $publisher=$b->getField("publisher");
    $year=$b->getField("year");
    $keywords=$b->getField("keywords");
    $fpath=$b->getField("fpath");
    $ftype=$b->getField("ftype");
    $dbq = "insert into bookshelf_booklist (author,title,publisher,year,keywords,".
           "fpath,ftype) values ('$author','$title','$publisher',$year,'$keywords',".
           "'$fpath','$ftype');";
    $db = DataBase::instance();
    $db->query($dbq);
  }
```

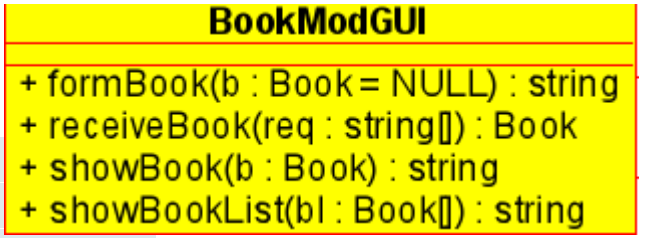
BookModDB – implementacja c.d.

```
public function getBookList() {
    $bl = new BookList();
    $db = DataBase::instance();
    $res=$db->getAll("select * from bookshelf_booklist;");
    foreach($res as $r) $bl->add(new Book($r));
    return $bl;
  }

  public function deleteBook($id) {
    $db = DataBase::instance();
    $res = $db->getRow("select * from bookshelf_booklist where id_b=$id;");
    if($res==FALSE) throw new Exception("InvalidBookId");
    $db->query("delete from bookshelf_booklist where id_b=$id;");
  }
```

BookModDB – implementacja c.d.

```
public function updateBook(Book $b) {
    $id = $b->getId();
    $author=$b->getField("author");
    $title=$b->getField("title");
    $publisher=$b->getField("publisher");
    $year=$b->getField("year");
    $keywords=$b->getField("keywords");
    $fpath=$b->getField("fpath");
    $ftype=$b->getField("ftype");
    $dbq = "update bookshelf_booklist ".
           "set author='$author',title='$title',publisher='$publisher',".
           "year=$year,keywords='$keywords',fpath='$fpath',ftype='$ftype'".
           "where id_b=$id;";
    $db = DataBase::instance();
    $db->query($dbq);
  }
```

BookModGUI




```
class BookModGUI {
  public function formBook(Book $b=NULL) {
  }

  public function receiveBook($req) {
  }


  public function showBook(Book $b) {
  }

  public function showBookList(BookList $bl) {
  }

};
```

UNIT TESTS - BookModUI

- testFormBookEmpty
- testFormBookUpdate
- testReceiveBookNew
- testReceiveBookUpdate
- testShowBookViewMode
- testShowBookEditMode
- testShowBookList

BookModGUI - implementacja

```
public function formBook(Book $b=NULL) {
    $id        = ($b==NULL) ? -1 :($b->getId());
    $author    = ($b==NULL) ? "" :($b->getField("author"));
    $title     = ($b==NULL) ? "" :($b->getField("title"));
    $publisher = ($b==NULL) ? "" :($b->getField("publisher"));
    $year      = ($b==NULL) ? "" :($b->getField("year"));
    $keywords  = ($b==NULL) ? "" :($b->getField("keywords"));
    $fpath     = ($b==NULL) ? "" :($b->getField("fpath"));
    $ftype     = ($b==NULL) ? "" :($b->getField("ftype"));

    $str = "<h1>Add or update book info</h1>";
    $str.= "<form action=".$_SERVER['PHP_SELF']." method=\"GET\">";
    $str.= "<input type=\"hidden\" name=\"cmd\" value=\"br\">";
    $str.= "<input type=\"hidden\" name=\"id_b\" value=\"$id\">";
    $str.= "Author: <input type=\"text\" name=\"author\" value=\"$author\"><br>";
    $str.= "Title: <input type=\"text\" name=\"title\" value=\"$title\"><br>";
    $str.= "Publisher: <input type=\"text\" name=\"publisher\" value=\"$publisher\"><br>";
    $str.= "Year: <input type=\"text\" name=\"year\" value=\"$year\"><br>";
    $str.= "Keywords: <input type=\"text\" name=\"keywords\" value=\"$keywords\"><br>";
    $str.= "File: <input type=\"text\" name=\"fpath\" value=\"$fpath\"><br>";
    $str.= "File type: <input type=\"text\" name=\"ftype\" value=\"$ftype\"><br>";
    $str.= "<input type=\"submit\" value=\"Save\"></form>";
    return $str;
}
```

BookModGUI – implementacja c.d.

```
public function receiveBook($req) {
    $b = new Book();
    $b->setId($req["id_b"]);
    $b->setField("author",$req["author"]);
    $b->setField("title",$req["title"]);
    $b->setField("publisher",$req["publisher"]);
    $b->setField("year",$req["year"]);
    $b->setField("keywords",$req["keywords"]);
    $b->setField("fpath",$req["fpath"]);
    $b->setField("ftype",$req["ftype"]);
    return $b;
  }
```

BookModGUI – implementacja c.d.

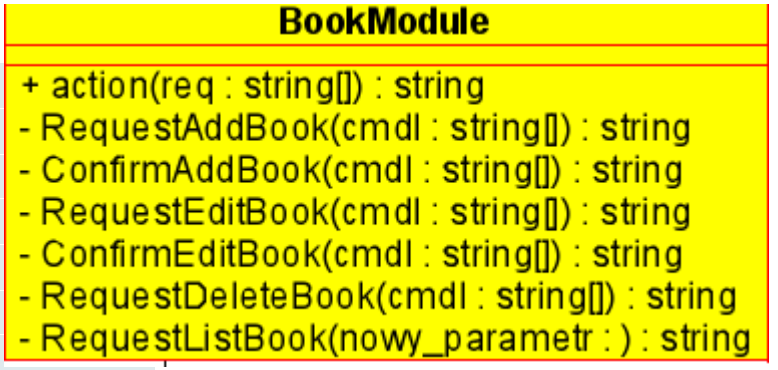
```
public function showBook(Book $b) {
    $str = "Book ID: ".$b->getId()."<br>";
    $str.= "Author: ".$b->getField("author")."<br>";
    $str.= "Title: ".$b->getField("title")."<br>";
    $str.= "Publisher (year): ".$b->getField("publisher").
        " (".$b->getField("year").") "."<br>";
    $str.= "Keywords: ".$b->getField("keywords")."<br>";
    $str.= "File: ".$b->getField("fpath")."<br>";
    $edit = AccMode::isEditMode();
    if($edit) {
        $id = $b->getId();
        $str.= "<table><tr><td>".
               HtmlTools::formButton("Edit",array("cmd"=>"be","id_b"=>"$id")).
               "</td><td>".
               HtmlTools::formButton("Delete",array("cmd"=>"bd","id_b"=>"$id")).
               "</td></tr></table>";
    };
    return $str;
  }
```

BookModGUI – implementacja c.d.

```
public function showBookList(BookList $bl) {
    $str = "<h1>Book list</h1>";
    $str.="<table border width=\"80%\">";
    foreach($bl as $b) $str.="<tr><td>".self::showBook($b)."</td></tr>";
    $str.="</table>";
    return $str;
  }
```

BookModule

```
class BookModule implements CommandInt {
  public function action($req) {
  }

  private function requestAddBook() {
  }

  private function receiveBook($req) {
  }

  private function requestEditBook($req) {
  }

  private function requestDeleteBook($req) {
  }

  private function requestListBook() {
  }

};
```



UNIT TESTS - BookModUI

testActionRequestAddBook

testActionReceiveNewBook

testActionReceiveUpdatedBook

testActionRequestEditBook

testActionRequestDeleteBook

testActionRequestListBook

BookModule - implementacja

```
class BookModule implements CommandInt {
  public function action($req) {
    if($req["cmd"]=="ba") return self::requestAddBook();
    if($req["cmd"]=="br") return self::receiveBook($req);
    if($req["cmd"]=="be") return self::requestEditBook($req);
    if($req["cmd"]=="bd") return self::requestDeleteBook($req);
    if($req["cmd"]=="bl") return self::requestListBook();
    return "";
  }

  private function requestAddBook() {
    return BookModGUI::formBook();
  }

  private function receiveBook($req) {
    $b = BookModGUI::receiveBook($req);
    if($b->getId()==-1) BookModDB::addBook($b);
    else BookModDB::updateBook($b);
    return self::requestListBook();
  }
```

BookModule – implementacja c.d.

```
private function requestEditBook($req) {
    $b = BookModDB::getBook($req["id_b"]);
    return BookModGUI::formBook($b);
  }

  private function requestDeleteBook($req) {
    BookModDB::deleteBook($req["id_b"]);
    return self::requestListBook();
  }

  private function requestListBook() {
    $bl = BookModDB::getBookList();
    return BookModGUI::showBookList($bl);
  }
```

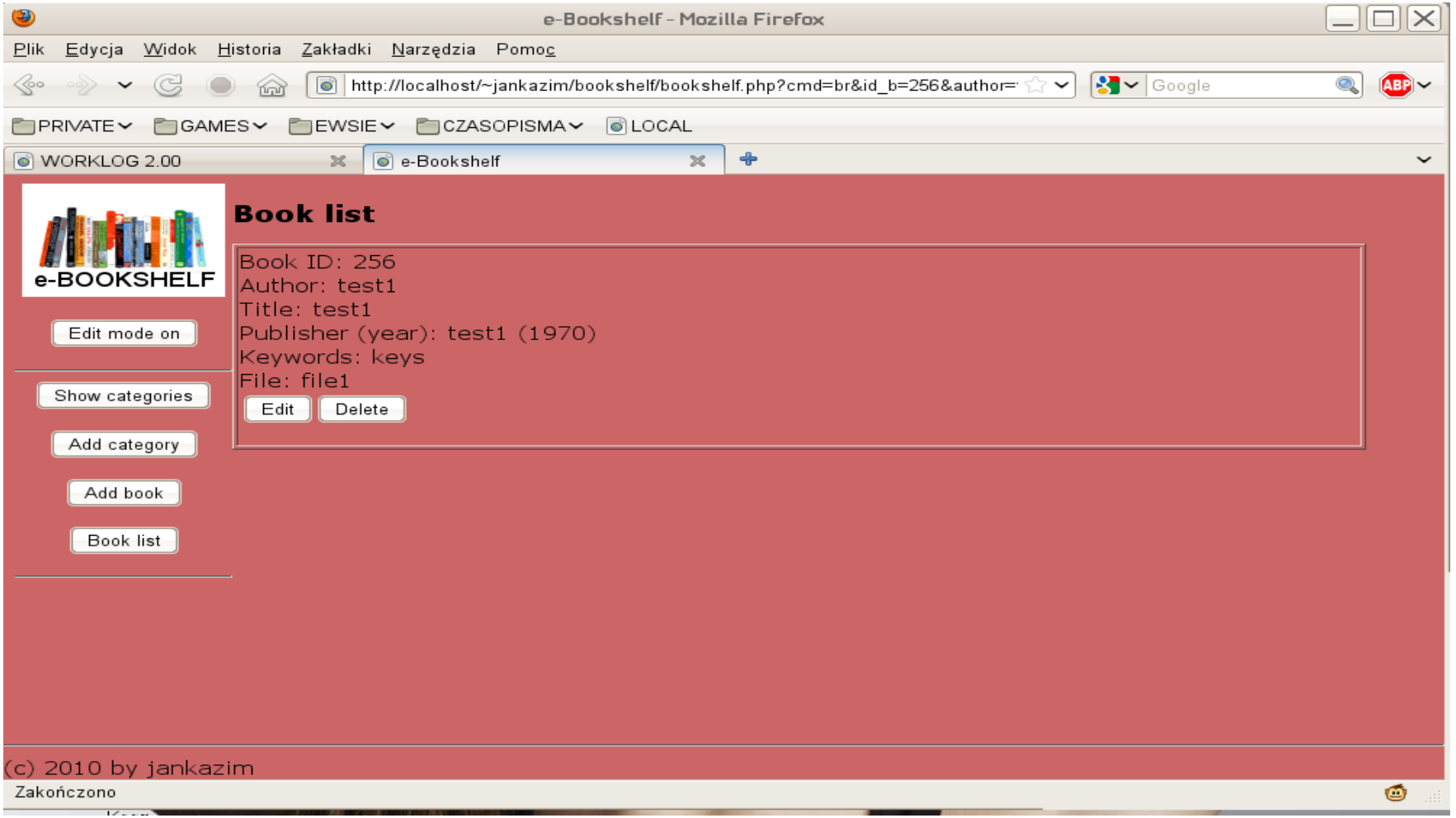
Dodanie przycisków obsługi modułu książek do menu (HtmlTools)

```
public function menu() {
    $str = "<hr>";
    $str.= self::formButton("Show categories",array("cmd"=>"cl"));
    $str.= self::formButton("Add category",array("cmd"=>"ca"));
    $str.= self::formButton("Add book",array("cmd"=>"ba"));
    $str.= self::formButton("Book list",array("cmd"=>"bl"));
    $str.= "<hr>";
    return $str;
  }
```

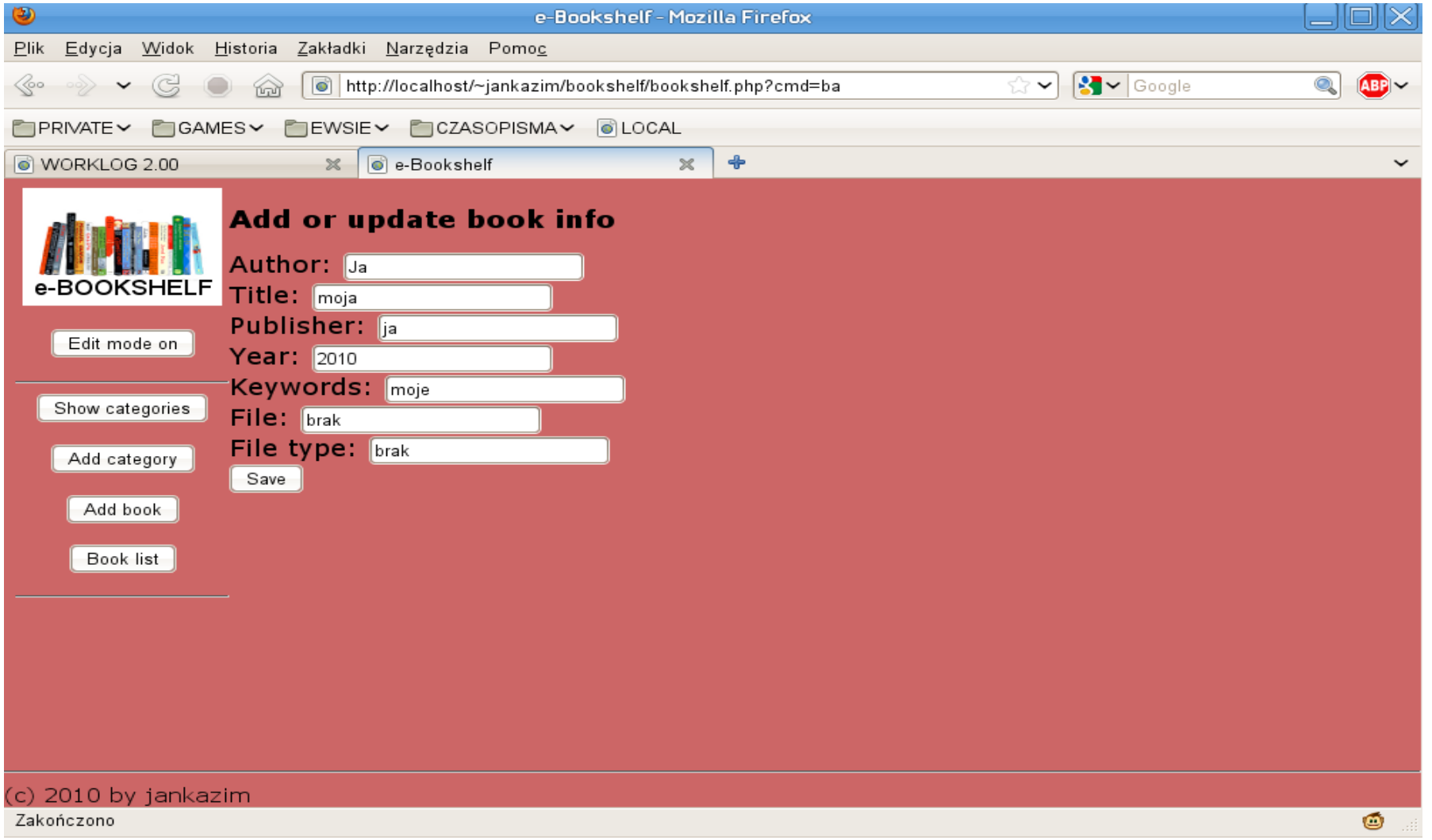
Włączenie do programu głównego modułu obsługi książek.

```
// ...

$data = $_REQUEST;

AccMode::setMode($data);
$rpan = AccMode::getModeButton();

$cm = new CatModule();
$bm = new BookModule();

if(!isset($data["cmd"])) $data["cmd"]="";
$content = $cm->action($data).$bm->action($data);

HtmlTools::page($rpan,$content);




?>
```

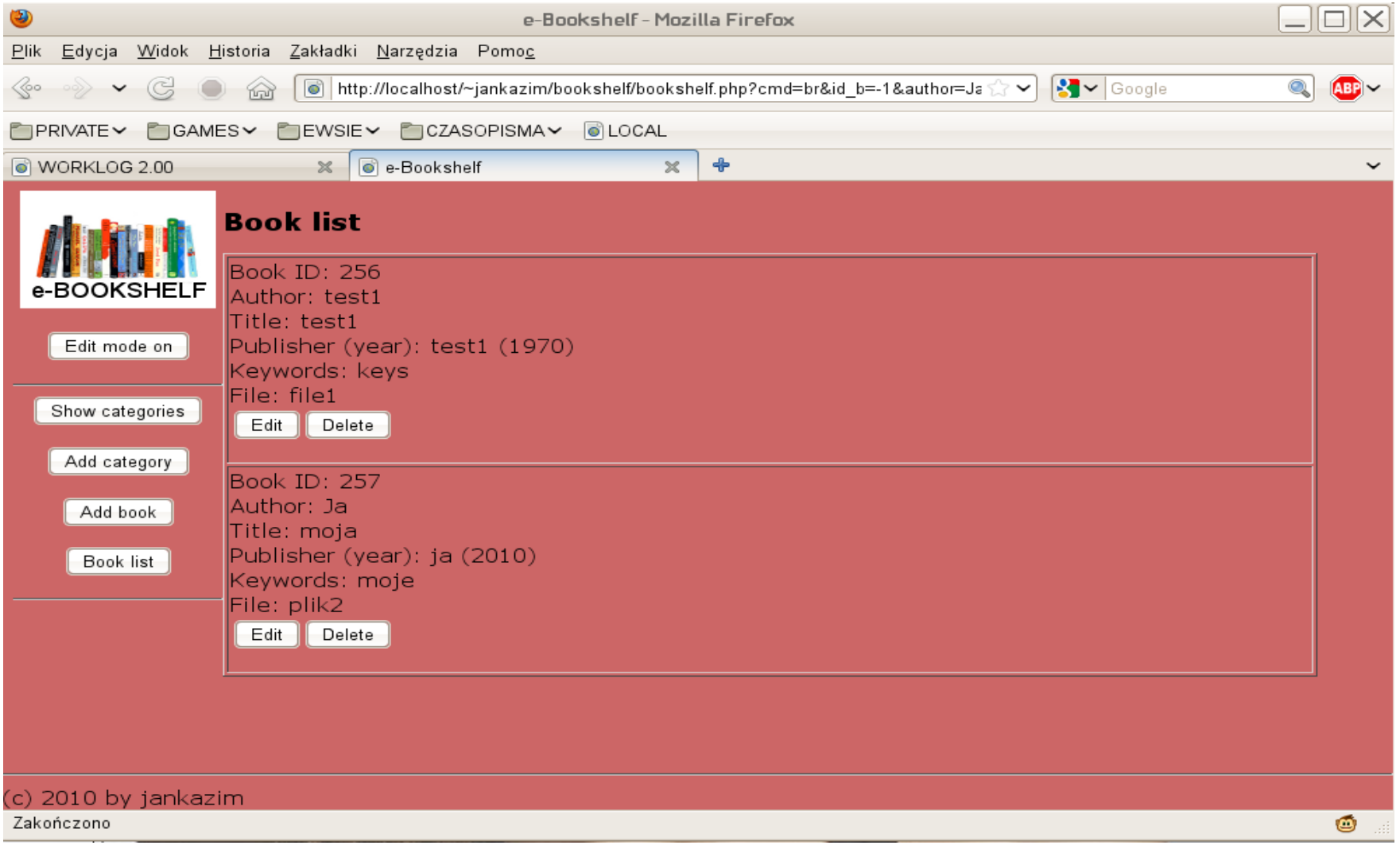
Moduł książek – działające GUI



Moduł książek – działające GUI c.d.



Moduł książek – działające GUI c.d.



TODO

- Refakoryzacja kodu – ujednolicić nazwy, usunąć duplikaty, wprowadzić klasy generyczne.
- Dodać obsługę błędów (w tej chwili brak).
- Namówić kogoś na zaprojektowanie GUI i trzymać się od tego z bardzo daleka.
- Uporządkować testy jednostkowe, wyczyścić kod.

Trochę statystyk

- Zrealizowane przypadki użycia 10 z 15, w tym:
  - niepełny przypadek użycia („lista książek”) zamiast szukaj książki (według podanych kryteriów).
  - niepełny przypadek „dodaj książkę” - brak możliwości przypisywania do kategorii. Należy rozważyć osobny przypadek użycia „przypisz książkę do kategorii”.

Trochę statystyk c.d.

- Rozmiar projektu:
  - liczba plików : 12
  - linii kodu : 1051
  - w tym na testy jednostkowe : 576
  - czas realizacji ok. 20 h

Dziękuję za udział w zajęciach i życzę powodzenia w realizacji własnych projektów