

Politechnika Śląska
Wydział Matematyki Stosowanej
Kierunek Informatyka
Studia stacjonarne I stopnia

Projekt inżynierski

System obsługi biblioteki naukowej

Kierujący projektem:
dr inż. Zdzisław Sroczyński

Autorzy:
Karolina Chrząszcz
Szymon Górnioczek
Tomasz Kryg

Gliwice 2018

*To jest
dedykacja*

Projekt inżynierski:

System obsługi biblioteki naukowej

kierujący projektem: dr inż. Zdzisław Sroczyński

1. Karolina Chrząszcz – (33%)

Zaprojektowanie i wykonanie aplikacji serwerowej oraz mobilnej.

2. Szymon Górnioczek – (33%)

Zaprojektowanie i wykonanie aplikacji serwerowej oraz strony internetowej dla administratora.

3. Tomasz Kryg – (33%)

Zaprojektowanie i wykonanie strony administratorskiej oraz mobilnej.

Podpisy autorów projektu

1.
2.
3.

Podpis kierującego projektem

.....

Oświadczenie kierującego projektem inżynierskim

Potwierdzam, że niniejszy projekt został przygotowany pod moim kierunkiem i kwalifikuje się do przedstawienia go w postępowaniu o nadanie tytułu zawodowego: inżynier.

Data

Podpis kierującego projektem

Oświadczenie autorów

Świadomy/a odpowiedzialności karnej oświadczam, że przedkładany projekt inżynierski na temat:

System obsługi biblioteki naukowej

został napisany przez autorów samodzielnie.

Jednocześnie oświadczam, że ww. projekt:

- nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2000 r. Nr 80, poz. 904, z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
- nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.
- nie zawiera fragmentów dokumentów kopiowanych z innych źródeł bez wyraźnego zaznaczenia i podania źródła.

Podpisy autorów projektu

1. Karolina Chrząszcz,
2. Szymon Górnioczek,
3. Tomasz Kryg,

nr albumu:225156,(podpis:)

nr albumu:225164,(podpis:)

nr albumu:225183,(podpis:)

Gliwice, dnia

Spis treści

Wstęp	7
1. Aplikacja serwerowa	9
1.1. Użyte frameworki	9
1.1.1. Modele Django	10
1.2. Model danych	10
1.2.1. Model Book	12
1.2.2. Model Category	13
1.3. Ważne klasy	13
1.3.1. Klasa CategoryTree	13
1.3.2. Klasa Dictionary	14
1.3.3. Klasa query	15
1.4. Usługi	17
1.4.1. Usługa dostarczająca książki	17
1.4.2. Usługa dostarczająca kategorie	18
1.4.3. Usługa dostarczająca słownik	19
1.5. Wdrażanie	20
1.5.1. Wirtualne środowisko	20
1.5.2. Pobranie aplikacji	20
1.5.3. Konfiguracja aplikacji	21
1.5.4. Uruchomienie aplikacji	22
1.5.5. Konfiguracja Cron	24
2. Panel administracyjny	25
2.1. Biblioteka	26
2.1.1. Importuj csv	26
2.1.2. Kategorie	27
2.1.3. Książki	27
2.2. Uwierzytelnianie i Autoryzacja	29
Literatura	31

Wstęp

wstęp jest tu

1. Aplikacja serwerowa

Aplikacja serwerowa LibraryApp, stanowi backend systemu bibliotecznego, oraz zapewnia narzędzia dla bibliotekarza i administratora systemu, pozwalające na zarządzanie aplikacją. Głównymi zadaniami tej aplikacji są:

- zarządzanie użytkownikami i ich uprawnieniami,
- zarządzanie bazą danych,
- zarządzanie informacją na temat woluminów biblioteki wydziałowej poprzez panel administracyjny,
- integracja aplikacji mobilnych poprzez usługę, dającą dostęp do informacji w bazie danych.

1.1. Użyte frameworki

Do stworzenia aplikacji został wykorzystany *Django*. Framework ten dostarcza rozwiązania takie jak:

- system autoryzacji użytkowników [1],
- możliwość zdefiniowania modelu danych kodem pythonowym oraz ORM wysokiego poziomu [2],
- automatycznie generowany i kompletny panel administracyjny [3],
- narzędzie do tworzenia serwisów webowych - *Django REST framework* [4],
- wsparcie dla wielojęzycznych aplikacji (w tym język polski) [5],

które kompletnie pokrywają potrzeby aplikacji LibraryApp. Oprócz tego, jak można przeczytać na oficjalnej stronie *Django* [7], framework jest zaprojektowany w taki sposób, aby robić częste zadania web-deweloperskie szybko i prosto. W praktyce oznacza to, że powstaje mało kodu aplikacji, dzięki czemu będzie on łatwiejszy

do zrozumienia i modyfikowania. Obszerna i aktualna dokumentacja, dostępna na stronie projektu, oraz duża społeczność korzystająca z *Django* sprawia, że rozwój aplikacji jest łatwiejszy.

1.1.1. Modele Django

W *Django* modele danych definiuje się jako klasy Python, które dziedziczą po `django.db.models.Model` [6]. Na podstawie takiej klasy *Django* automatycznie generuje tabele w bazie danych, każdy atrybut modelu reprezentuje pole w tabeli. Przykładowo, z modelu zdefiniowanego w ten sposób:

```
class Category(models.Model):
    category_id = models.CharField(
        max_length=200,
        unique=True,
        verbose_name='Id kategorii'
    )
    category_name = models.TextField(
        verbose_name='Nazwa kategorii'
    )
```

zostanie wygenerowana tabela w bazie danych:

```
CREATE TABLE 'libraryapp_category' (
'id'integer NOT NULL PRIMARY KEY AUTOINCREMENT,
'category_id'varchar(200) NOT NULL UNIQUE,
'category_name'text NOT NULL
);
```

1.2. Model danych

Model danych został zaprojektowany na podstawie specyfikacji:

- arkusza kalkulacyjnego z danymi (`dane-prog-bibl.xlsx`),
- dokumentu zawierającego opis pól (`BIBLIOTEKA-program.docx`).

Arkusz `dane-prog-bibl.xlsx` składa się z 3 zakładek:

- KSIAZKA,
- KATEGORIE,
- PRZYPISANIE_KATEGORII.

Zakładka KSIAZKA składa się z kolumn:

- SYG_MS,
- SYG_BG
- OZN_OPDOW,
- TYTUL,
- TOM,
- ROK,
- ISBN/ISSN,
- TYP,
- DOSTEPNOSC.

Zakładka KATEGORIE składa się z kolumn:

- ID_KATEGORII,
- KATEGORIA.

Zakładka PRZYPISANIE_KATEGORII składa się z kolumn:

- SYG_MS,
- ID_KATEGORII.

Na podstawie takiej struktury danych zostały zaprojektowane dwa modele:

- `Book`, odpowiadający danym z zakładek KSIAZKA i PRZYPISANIE_KATEGORII,
- `Category`, odpowiadający danym z zakładki KATEGORIE.

Wszystkie definicje modeli znajdują się w pliku `/libproject/libraryapp/models.py`.

1.2.1. Model Book

Model danych Book jest zdefiniowany jako model *Django*, składa się z pól:

- `signature_ms`,
typu: `IntegerField`,
odpowiada danym z kolumny: SYG_MS,
- `signature_bg`,
typu: `CharField`,
odpowiada danym z kolumny: SYG_BG,
- `responsibility`,
typu: `TextField`,
odpowiada danym z kolumny: OZN_OPDOW,
- `title`
typu: `CharField`,
odpowiada danym z kolumny: TYTUL,
- `volume`
typu: `CharField`,
odpowiada danym z kolumny: TOM,
- `year`
typu: `IntegerField`,
odpowiada danym z kolumny: ROK,
- `isbn_issn`
typu: `CharField`,
odpowiada danym z kolumny: ISBN/ISSN,
- `type`
typu: `CharField`,
odpowiada danym z kolumny: TYP,
przyjmuje wartości: 'podręcznik', 'wzrostkowy', 'zbiór zadań',
- `availability`
typu: `CharField`,

odpowiada danym z kolumny DOSTEPNOSC,
przyjmuje wartości: 'dostępna', 'wypożyczona', 'czytelnia'.

- **categories**
typu: `ManyToManyField`,
odpowiada danym z zakładki PRZYPISANIE_KATEGORII.

1.2.2. Model Category

Model danych `Category` jest zdefiniowany jako model *Django*, składa się z pól:

- **category_id**,
typu: `CharField`,
odpowiada danym z kolumny: `ID_KATEGORII`,
- **category_name**,
typu: `TextField`,
odpowiada danym z kolumny: `KATEGORIA`.

1.3. Ważne klasy

Oprócz modeli opracowanych na podstawie dostarczonych danych, zostały zaprojektowane jeszcze klasy takie jak: `CategoryTree`, `Dictionary`, `Query`.

1.3.1. Klasa `CategoryTree`

Ze względu na wymagania opisane w punktach 10 *Kategoria główna* i 11 *kategoria szczegółowa*, dokumentu `BIBLIOTEKA-program.docx`, dotyczące podziału kategorii na "kategoria główna" i "kategoria szczegółowa", została zaimplementowana klasa która przedstawia tę relację między kategoriami. Kategoria główna to taka, której id jest postaci "G_x", gdzie x to ciąg cyfr, na przykład **G_00:ALGEBRA**. Kategoria główna może mieć kategorie szczegółowe, których id jest postaci "G_x-S_y", gdzie x i y to ciągi cyfr, część znajdująca się przed "-" to id kategorii głównej. Zatem kategoria **G_00-S_04:algebry Boole'a**, jest kategorią szczegółową kategorii **G_00:ALGEBRA**.

Klasa `CategoryTree` składa się z pól:

- `main_category`, zawiera obiekt `Category`, odpowiadający kategorii głównej,
- `subcategories`, zawiera tablicę obiektów `Category`, odpowiadającym kategoriom szczegółowym.

Przykład:

```
{
  "main_category": {
    "category_id": "G_22",
    "category_name": "TOPOLOGIA"
  },
  "subcategories": [{
    "category_id": "G_22-S_00",
    "category_name": "topologia ogólna"
  },
  {
    "category_id": "G_22-S_01",
    "category_name": "topologia algebraiczna"
  }
]
```

1.3.2. Klasa `Dictionary`

W celu dostarczenia do aplikacji mobilnych informacji o wartościach jakie mogą przyjmować niektóre pola w bazie danych oraz ułatwić w przyszłości modyfikowanie zakresu tych wartości, została zaimplementowana klasa `Dictionary`. Obiekt tej klasy zawiera składa się z pól:

- `types`, tablica zawierająca wartości jakie może przyjąć pole `Book.types`,
- `availability_types`, tablica zawierająca wartości jakie może przyjąć pole `Book.availability`.

```
{
  "availability_types": [
```

```
        "dostępna",
        "wypożyczona",
        "czytelnia"
    ],
    "types": [
        "podręcznik",
        "inny",
        "zbiór zadań"
    ]
}
```

1.3.3. Klasa query

Obiekt klasy `query` definiuje kryteria, według których, serwis zwróci książki. Klasa `query` składa się z pól:

- `filters` - Obiekt definiujący filtry pól książek. Pola obiektu `filters` są opcjonalne. Aby otrzymać książki których tytuł jest równy pewnej wartości, obiekt `filters` należy zdefiniować w następujący sposób:

```
"filters":{
    "title":"wartość"
}
```

Aby otrzymać z serwisu książki, których tytuł zawiera jakiś literał, obiekt `filters` należy zdefiniować w następujący sposób:

```
"filters":{
    "title__contains":"wartość"
}
```

Filtry pozostałych pól definiujemy analogicznie. Filtrowanie po wielu polach zdefiniowane jest w następujący sposób:

```
"filters":{
```

```
    "title__contains": "wartość",
    "responsibility__contains": "GŁ",
    "year": 2003
}
```

- **categories** - tablica id kategorii, zwrócone książki zawierają jedną z kategorii podanych w tablicy. Przykład:

```
"categories": ["G_00", "G_01-S23"]
```

Aby otrzymać z serwisu książki, które posiadają konkretny zestaw kategorii, należy zdefiniować obiekt **filter**:

```
"filters": {
    "categories": ["G_00", "G_01-S23"]
}
```

- **pagination** - klasa pozwalająca na zdefiniowanie otrzymanego wycinka danych, zapobiega to zwróceniu zbyt dużej liczby danych jednocześnie. Obiekt zawiera pola **offset** - definiuje ile pierwszych pozycji odrzucić, **limit** - definiuje ile pozycji może zostać zwróconych maksymalnie, podczas jednego zapytania.

Przykładowy obiekt **query**:

```
"query": {
    "filters": {
        "responsibility__contains": "GŁ",
        "availability": "dostępna"
    },
    "categories": ["G_00"],
    "pagination": {
        "offset": 20,
        "limit": 10
    }
}
```

1.4. Usługi

Aplikacja serwerowa zapewnia usługi dające dostęp do informacji zawartych w bazie danych. Usługi dostępne są za pomocą REST Api na środowisku uczelnianym pod adresem 157.158.16.217:8000. Do korzystania z usług nie jest wymagana autoryzacja.

1.4.1. Usługa dostarczająca książki

Usługa ta zwraca listę książek (book), według kryterium zdefiniowanego przez obiekt query.

Zapytanie:

- Enpoint: /books,
- Method: POST,
- Headers: "Content-Type": "application/json",
- Body: JSON (application/json): query,

Odpowiedź:

- Tablica obiektów Book.

Przykład odpowiedzi:

```
[{
  "signature_ms": 601,
  "signature_bg": "",
  "responsibility": "JAGŁOM I.M.; BOŁTIAŃSKI W.G.",
  "title": "Figury wypukłe",
  "volume": "",
  "year": 1955,
  "isbn_issn": "",
  "type": "inny",
  "availability": "dostępna",
  "categories": [{
    "category_id": "G_05-S_03",
```

```
        "category_name": "geometryczne pojęcie wypukłości"
    }
}]
```

1.4.2. Usługa dostarczająca kategorie

Usługa ta zwraca listę wszystkich kategorii przedstawionych jako obiekt `CategoryTree`.
Zapytanie:

- Enpoint: `/categories`,
- Method: `GET`,
- Headers: `"Content-Type": "application/json"`,

Odpowiedź:

- Tablica obiektów `CategoryTree`.

Przykładowa odpowiedź:

```
[{
  "main_category": {
    "category_id": "G_00",
    "category_name": "ALGEBRA"
  },
  "subcategories": [{
    "category_id": "G_00-S_00",
    "category_name": "algebra matematyczna"
  }]
},
{
  "main_category": {
    "category_id": "G_01",
    "category_name": "ANALIZA"
  },
  "subcategories": [{
    "category_id": "G_01-S_00",
```

```
        "category_name": "analiza matematyczna"
    }]
}
]
```

1.4.3. Usługa dostarczająca słownik

Zwraca obiekt `Dictionary` zawierający wszystkie typy książek oraz typy ich dostępności.

Zapytanie:

- Endpoint: `/dictionary`,
- Method: `GET`,
- Headers: `"Content-Type": "application/json"`,

Odpowiedź:

- Obiekt `Dictionary`.

Przykładowa odpowiedź:

```
{
  "types": [
    "podręcznik",
    "inny",
    "zbiór zadań"
  ],
  "availability_types": [
    "dostępna",
    "wypożyczona",
    "czytelnia"
  ]
}
```

1.5. Wdrażanie

Aby uruchomić aplikację wymagany jest jedynie zainstalowany *python3.6*. Aplikację można uruchomić w systemach Linux, OS X i Windows [8, s. 140], ten rozdział jednak opisuje proces wdrażania w systemie Linux. Skrypty załączone razem z aplikacją (`setup_app.sh` oraz `boot_script.sh`), są skryptami bashowymi, działającymi pod systemem Linux.

1.5.1. Wirtualne środowisko

Wszystkie zależności potrzebne do uruchomienia aplikacji są instalowane wewnątrz wirtualnego środowiska, dzięki czemu proces wdrażania można w dużej części zautomatyzować. Aby móc stworzyć wirtualne środowisko należy zainstalować paczkę *virtualenv* korzystając z polecenia:

```
sudo pip3 install virtualenv
```

1.5.2. Pobranie aplikacji

Aplikacja jest dostępna na repozytorium `github.com/szymongor/library`, aby ją pobrać można skorzystać z *git* lub pobrać paczkę w formie *zip* i rozpakować na maszynie, zalecane jest jednak użycie *gita*, ułatwi to w przyszłości nanoszenie zmian w aplikacji. Szczegóły dotyczące instalacji *gita* dostępne są na stronie `git-scm.com/book/en/v2/Getting-Started-Installing-Git/`.

Aby ściągnąć aplikację poprzez *git* należy skorzystać z komendy:

```
git clone https://github.com/szymongor/library.git
```

w rezultacie, powinien się pojawić folder `library` w obecnej lokalizacji. Należy otworzyć folder z aplikacją używając komendy `cd library`. Katalog powinien zawierać:

- `boot_script.sh` - skrypt bashowy, uruchamiający aplikację,
- `librarysite` - folder zawierający pliki aplikacji *Django*,
- `properties.py` - plik zawierający konfigurację aplikacji,
- `setup_app.sh` - skrypt bashowy, automatycznie konfiguruje środowisko,

- `libraryapp` - folder z plikami aplikacji Django,
- `manage.py` - skrypt *Django*, do zarządzania aplikacją,
- `requirements.txt` - lista zależności potrzebna do uruchomienia aplikacji, zostanie automatycznie załadowana przez skrypt `setup_app.sh`,
- `static` - folder z plikami statycznymi aplikacji.

1.5.3. Konfiguracja aplikacji

Aby skonfigurować aplikację należy edytować plik `properties.py`, np używając polecenia:

```
nano properties.py
```

Konfiguracja zastosowana na środowisku Politechniki Śląskiej:

```
PROJECT_PATH="/home/biblio/libproject/"
VENV="libenv"
HOST="157.158.16.217:8000"
LOG_FILE="logs"
SECRET_KEY='f1b($dr1^t$8!#$#!a^8xmj0+#$pnaramf1r^xd(t5a%ghjai_\'
ALLOWED_HOSTS=['127.0.0.1','localhost','157.158.16.217']
USER_NAME="BibliotekaMS"
PASSWORD="tajne_haslo_nalezy_zmienic"
MAIL="admin@mail.com"
```

Plik konfiguracyjny składa się z pól:

- `PROJECT_PATH (!)` - lokalizacja folderu library z aplikacją,
- `VENV (*)` - nazwa wirtualnego środowiska, jeśli jeszcze nie zostało utworzone, skrypt `setup_app.sh` automatycznie utworzy wirtualne środowisko o takiej nazwie,
- `HOST (!)` - adres IP oraz port na które zostanie wystawiona aplikacja,
- `LOG_FILE (*)` - nazwa pliku z logami aplikacji,

- **SECRET_KEY (!)**- tajny klucz, typu string, długości 50, używany przez *Django* np do podpisywania plików cookies i autoryzacji,
- **ALLOWED_HOSTS (!)**- tablica adresów hostów/domen, które mogą serwować aplikację, zabezpieczenie ze strony *Django* przed atakami typu Host Header Injection,
- **USER_NAME (*)**- nazwa użytkownika panelu administracyjnego, mającego uprawnienia administratorskie w aplikacji,
- **PASSWORD (!)**- hasło użytkownika,
- **MAIL (*)**- adres email użytkownika.

Pola oznaczone "!" należy edytować, pola oznaczone "*" można edytować lub pozostawić niezmienione.

1.5.4. Uruchomienie aplikacji

Podczas pierwszego uruchomienia aplikacji należy wywołać komendę:

```
./setup_app.sh
```

z poziomu głównego folderu aplikacji (**library**). Komenda ta uruchomi skrypt, który realizuje takie zadania jak:

- utworzenie wirtualnego środowiska o nazwie zdefiniowanej w pliku **properties.py** jako **VENV**,
- instalacja zależności wypisanych w pliku **requirements.txt** wewnątrz wirtualnego środowiska,
- utworzenie konta administratora aplikacji, zgodnie z danymi podanymi w **properties.py** (**USER_NAME**, **PASSWORD**, **MAIL**),
- wygenerowanie bazy danych na podstawie modelu zdefiniowanego w pliku **libraryapp/models.py**.

Aby uruchomić aplikację należy wywołać:

```
./boot_script.sh
```

jeśli skrypt został wykonany poprawnie powinien się ukazać komunikat podobny do tego:

```
Django version 2.0, using settings 'librarysite.settings'
Starting development server at http://157.158.16.217:8000/
Quit the server with CONTROL-C.
```

a po wpisaniu w przeglądarkę adresu hosta (w tym przypadku 157.158.16.217:8000), powinno się ukazać okno logowania do panelu administracyjnego.

W ten sposób uruchomiona aplikacja zostanie wyłączona po zamknięciu terminala. Aby uruchomić aplikację w tle jako proces niezależny od terminala można skorzystać z wirtualnej konsoli, używając polecenia:

```
screen ./boot_script.sh
```

aby opuścić wirtualną konsolę należy użyć skrótu `ctrl+a` a następnie `d`. Po opuszczeniu konsoli wyświetli się komunikat:

```
[detached from identyfikator_konsoli]
```

Aby włączyć ponownie wirtualną konsolę z uruchomioną aplikacją należy użyć polecenia:

```
screen -r identyfikator_konsoli
```

Polecenie:

```
screen -ls
```

wyświetli wszystkie aktywne konsole.

Aby wyłączyć aplikację można przełączyć się na wirtualną konsolę z uruchomioną aplikacją i użyć skrótu: `ctrl+c`.

1.5.5. Konfiguracja Cron

Z powodu częstych przerw w działaniu serwera, konieczne jest uruchamianie aplikacji za każdym razem gdy serwer się uruchamia. Do tego celu został wykorzystany *Cron*, program do harmonogramowania zadań [9]. Aby skonfigurować *Crona*, by uruchamiał aplikację wraz z uruchomieniem serwera, należy użyć polecenia:

```
crontab -e
```

i w ostatniej linii pliku dodać wpis:

```
@reboot cd /home/<lokalizacja_projektu>/library && ./boot_script.sh
```

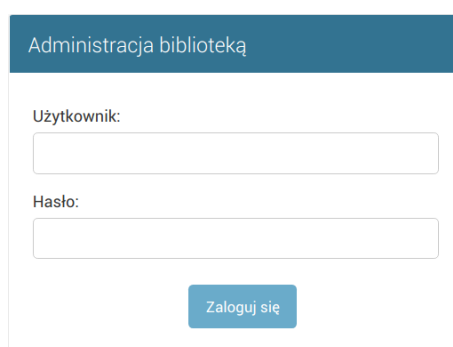
Aby przetestować wywołanie zadania, można zdefiniować zadanie wywołujące się co minutę w analogiczny sposób:

```
* * * * * cd /home/<lokalizacja_projektu>/library && ./boot_script.sh
```

po minucie powinna się uruchomić aplikacja, ważne aby po teście usunąć ten wpis z *Crona*.

2. Panel administracyjny

Panel administracyjny dostępny jest pod adresem 157.158.16.217:8000/admin. Po wpisaniu tego adresu w przeglądarce, powinno wyświetlić się okno logowania zatytułowane "Administracja biblioteką", przedstawione na rysunku 1. Dane do logowania, jeśli nie zostały wcześniej zmienione, powinny być takie jak w pliku `properties.py` (USER_NAME i PASSWORD).



The screenshot shows a login form titled "Administracja biblioteką". It contains two input fields: "Użytkownik:" (User) and "Hasło:" (Password). Below the fields is a blue button labeled "Zaloguj się" (Log in).

Rysunek 1: Okno logowania do panelu administracyjnego

Po zalogowaniu powinien się ukazać panel administracyjny do zarządzania biblioteką (rysunek 2). W nagłówku strony zatytułowanym "Administracja biblioteką", znajduje się odnośnik do zmiany hasła oraz wylogowania obecnego użytkownika. Poniżej nagłówka znajdują się dwie sekcje zatytułowane "Biblioteka" i "Uwierzytelnianie i Autoryzacja". Po prawej stronie sekcja "Ostatnie działania", wyświetlająca listę ostatnich zmian wprowadzonych w aplikacji.

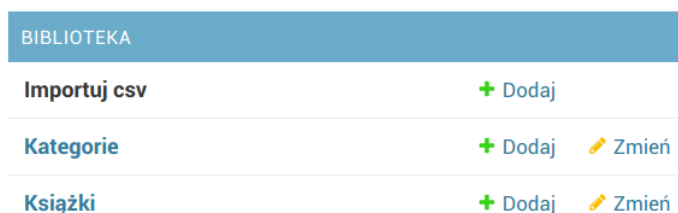



The screenshot shows the main dashboard of the library administration panel. The header is titled "Administracja biblioteką" and includes links for "WITAJ, ADMIN", "ZMIANA HASŁA", and "WYLOGUJ SIĘ". The main content area is divided into two sections: "Administracja zasobami" (Resource Management) and "Uwierzytelnianie i autoryzacja" (Authentication and Authorization). The "Administracja zasobami" section has a sub-header "BIBLIOTEKA" and lists "Importuj csv", "Kategorie", and "Książki", each with "Dodaj" (Add) and "Zmien" (Edit) buttons. The "Uwierzytelnianie i autoryzacja" section has a sub-header "Grupy" (Groups) with "Dodaj" and "Zmien" buttons. On the right side, there is a sidebar titled "Ostatnie działania" (Recent Actions) with a sub-header "Moje działania" (My Actions) and a list of actions, each with a yellow lightning bolt icon and the text "1 1 Książka".

Rysunek 2: Strona główna panelu administracyjnego

2.1. Biblioteka

W skład tej sekcji wchodzi opcje importowania plików CSV, zarządzania kategoriami i książkami.

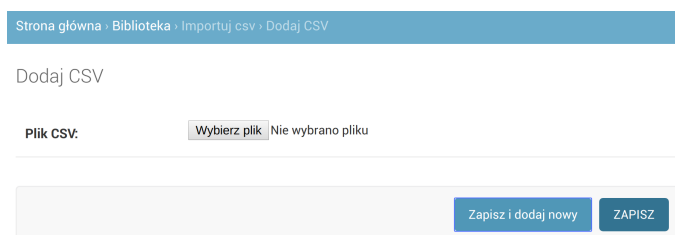


BIBLIOTEKA	
Importuj csv	+ Dodaj
Kategorie	+ Dodaj  Zmień
Książki	+ Dodaj  Zmień

Rysunek 3: Sekcja "Biblioteka"

2.1.1. Importuj csv

Po naciśnięciu "+ Dodaj", w wierszu "Importuj csv" (rysunek 3), otworzy się panel zatytułowany "Dodaj CSV", widoczny na rysunku 4. Aby wczytać plik CSV, należy wskazać jego lokalizację (klikając przycisk "Wybierz plik") a następnie kliknąć przycisk "Zapisz". Po załadowaniu danych z pliku CSV wyświetli się lista komunikatów dotycząca statusu importowanych danych.



Strona główna > Biblioteka > Importuj csv > Dodaj CSV

Dodaj CSV

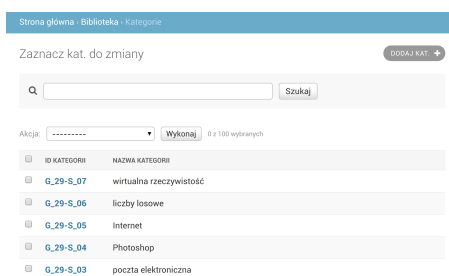
Plik CSV: Nie wybrano pliku

Rysunek 4: Formmularz importowania plików CSV

Pliki CSV powinny zostać wygenerowane na podstawie arkusza kalkulacyjnego `dane-prog-bibl.xlsx`, kodowanie pliku to UTF-8, symbol oddzielający pola to `,`, a separator tekstu to symbol `"`. Aplikacja sama rozpoznaje czy importowany plik CSV zawiera informacje z zakładki KSIAZKA, KATEGORIE czy PRZYPISANIE_KATEGORII.

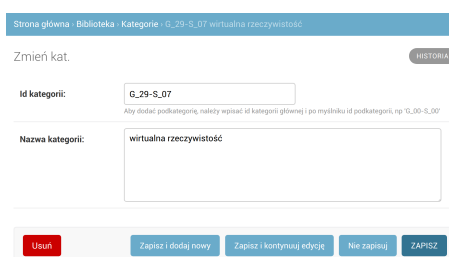
2.1.2. Kategorie

Wybór "Kategorie" w sekcji "Biblioteka" (rysunek 3), przenosi do widoku z listą wszystkich kategorii, widocznego na rysunku 5. Z poziomu tej listy można usuwać wybrane kategorie oraz wyszukiwać kategorie wpisując w pole tekstowe części id kategorii lub nazwy kategorii. Klikając na id kategorii w liście, otworzy się formularz edycji danej kategorii, widoczny na rysunku 6.



Rysunek 5: Widok listy kategorii

Po naciśnięciu "Dodaj kat. +" (prawy górny róg na rysunku 5), otworzy się formularz tworzenia nowej kategorii. Aby dodać kategorię główną nadajemy jej id "G_nr_kat", gdzie "nr_kat" to ciąg cyfr. Aby dodać kategorię szczegółową nadajemy jej id "G_nr_kat-S_nr_sub_kat", gdzie "G_nr_kat" to id kategorii głównej a "nr_sub_kat" to id kategorii szczegółowej. Z poziomu okna edycji kategorii można podejrzeć historię naniesionych zmian wybierając "Historia" (prawy górny róg rysunku 6).



Rysunek 6: Formularz edycji kategorii

2.1.3. Książki

Po naciśnięciu "Książki" w sekcji "Biblioteka" (rysunek 3), otworzy się widok z listą książek, widoczny na rysunku 7.

Akcja	SYGNATURA MS	SYGNATURA KO	OZNACZENIE ODPWIEDZIALNOŚCI	TYTUŁ	TOM	ROK	ISBN/ISSN	TYP	DOSTĘPNOŚĆ
<input checked="" type="checkbox"/>	15082	Z.13138	MISHRA Shashi Kant, UPADHYAY Balendu Bhoushan	Pseudolinear functions and optimization	-	2015	9781482255737	inny	czytelna
<input checked="" type="checkbox"/>	15081	Z.13137	POWERS Joseph M., SEN Mohi	Mathematical methods in engineering	-	2015	9781107037045	podręcznik	czytelna
<input checked="" type="checkbox"/>	15076	Z.13132	GUPTA Radhey S.	Elements of numerical analysis	-	2015	9781107004495	podręcznik	czytelna
<input type="checkbox"/>	14993		BRONSTEIN I.N., SEMENOV A.P., MUSKELIS G., MUSKELIS H.	Nonoscillatory kernels in mathematical analysis	2004	2004	8301141464	inny	czytelna
<input type="checkbox"/>	14992		BRONSTEIN I.N., SEMENOV A.P., MUSKELIS G., MUSKELIS H.	Nonoscillatory kernels in mathematical analysis	2004	2004	8301141464	inny	czytelna

Rysunek 7: Widok listy książek

Z poziomu widoku z listą można:

- wyszukiwać książki, wpisując jej id (całe) lub fragment tytułu,
- sortować wyświetlone książki po dowolnym zbiorze pól, klikając na nazwę tych pól w nagłówku tablicy, w odpowiedniej kolejności,
- usuwać wybrane książki, poprzez zaznaczenie zbioru książek i wykonanie akcji "Usuń wybrane Książki",
- wybrać książkę do edycji, klikając jej id na liście,
- dodać nową książkę, klikając "Dodaj Książka +".

Po wybraniu książki do edycji, otworzy się formularz edycji książki widoczny na rysunku 8.

Zmieni Książka

Sygnatura ms: 15082

Sygnatura ko: Z.13138

Oznaczenie odpowiedzialności: MISHRA Shashi Kant, UPADHYAY Balendu Bhoushan

Tytuł: Pseudolinear functions and optimization

Rysunek 8: Formularz edycji książki

Z poziomu okna edycji książki można przejrzeć historię naniesionych zmian wybierając "Historia". Widok zmian widoczny na rysunku 9.

Strona główna · Biblioteka · Książki · 15082 Pseudolinear functions and optimization · Historia		
Historia zmian: 15082 Pseudolinear functions and optimization		
DATA/CZAS	UŻYTKOWNIK	AKCJA
13 lutego 2018 17:29	admin	Żadne pole nie zostało zmienione.
13 lutego 2018 17:29	admin	Żadne pole nie zostało zmienione.
16 lutego 2018 16:27	admin	Zmodyfikowano type.

Rysunek 9: Widok historii zmian książki

2.2. Uwierzytelnianie i Autoryzacja

Sekcja ta składa się z opcji tworzenia i zarządzania kontami użytkowników i grup użytkowników.

UWIERZYTELNIANIE I AUTORYZACJA		
Grupy	+ Dodaj	Zmień
Użytkownicy	+ Dodaj	Zmień

Rysunek 10: Sekcja ”Uwierzytelnianie i Autoryzacja”

Obecnie aplikacja nie wykorzystuje grup użytkowników, opcja ta została pozostawiona gdyby w przyszłości projekt miałby zawierać konta studentów z możliwością rezerwacji książek. Aby dodać nowe konto administratora biblioteki należy wybrać ”+ Dodaj”, w wierszu ”Użytkownicy” (rysunek 10), następnie uzupełnić formularz widoczny na rysunku 11 i zapisać.

Strona główna · Uwierzytelnianie i autoryzacja · Użytkownicy · Dodaj użytkownika	
Dodaj użytkownik	
Najpierw podaj nazwę użytkownika i hasło. Następnie będziesz mógł edytować więcej opcji użytkownika.	
Użytkownik:	<input type="text"/> <small>Wymagana. 150 lub mniej znaków. Jedynie litery, cyfry i @./+!-/_</small>
Hasło:	<input type="password"/> <small>Twoje hasło nie może być zbyt podobne do twoich innych danych osobistych. Twoje hasło musi zawierać co najmniej 8 znaków. Twoje hasło nie może być powszechnie używanym hasłem. Twoje hasło nie może składać się tylko z cyfr.</small>
Potwierdzenie hasła:	<input type="password"/> <small>Wprowadź to samo hasło ponownie, dla weryfikacji.</small>
<div>Zapisz i dodaj nowyZapisz i kontynuuj edycjęZAPISZ</div>	

Rysunek 11: Formularz nowego użytkownika

Nowy użytkownik powinien się teraz pojawić na liście wszystkich użytkowników aplikacji. Aby nadać mu uprawnienia administracyjne, należy wybrać na liście (widocznej na rysunku 12) nowo dodanego użytkownika.

The screenshot shows a web interface for managing users. At the top, a blue header bar contains the text "Strona główna · Uwierzytelnianie i autoryzacja · Użytkownicy". Below this, a green notification bar states "Użytkownik „Admin_nowy” został pomyślnie zmieniony." The main area is titled "Zaznacz użytkownik do zmiany" and features a search bar with a magnifying glass icon and a "Szukaj" button. Below the search bar, there's a dropdown menu for "Akcja:" with a "Wykonaj" button and a counter "0 z 2 wybranych". A table lists users with columns: "UŻYTKOWNIK", "ADRES EMAIL", "IMIĘ", "NAZWISKO", and "W ZESPOLE". The table contains two rows: "Admin_nowy" (highlighted in blue) and "admin" (with email "admin@mail.com"). To the right of the table is a "FILTR" sidebar with two sections: "Używając w zespole" (with options "Wszystko", "Tak", "Nie") and "Używając status superużytkownika" (with options "Wszystko", "Tak"). A "DODAJ UŻYTKOWNIK +" button is located at the top right of the main area.

UŻYTKOWNIK	ADRES EMAIL	IMIĘ	NAZWISKO	W ZESPOLE
Admin_nowy				
admin	admin@mail.com			

Rysunek 12: Lista użytkowników aplikacji

W formularzu edycji nowego użytkownika (rysunek 13), w sekcji "Uprawnienia", należy zaznaczyć opcję "W zespole" i zapisać.

The screenshot shows the "Administracja biblioteką" header with links "WITAJ, ADMIN.", "ZMIANA HASŁA", and "WYLOGUJ SIĘ". Below is a breadcrumb "Strona główna · Uwierzytelnianie i autoryzacja · Użytkownicy · Admin_nowy". The main title is "Zmień użytkownik" with a "HISTORIA" button. The form has two main sections: "Użytkownik:" with a text input containing "Admin_nowy" and a note "Wymagana: 150 lub mniej znaków. Jedynie litery, cyfry i @/./+/-/_..."; and "Hasło:" with a note about password requirements and a "hash:" field. Below these is the "Uprawnienia" section with two checkboxes: "Aktywny" (checked) and "W zespole" (checked). Each checkbox has a descriptive text: "Określa czy użytkownika należy uważać za aktywnego. Odnazcz zamiast usuwać konto." and "Określa czy użytkownik może zalogować się do panelu admina."

Rysunek 13: Formularz edycji użytkownika

Literatura

- [1] <https://docs.djangoproject.com/en/2.0/ref/contrib/auth/> [dostęp: 10 lutego 2018]
- [2] <https://docs.djangoproject.com/en/2.0/topics/db/queries/> [dostęp: 10 lutego 2018]
- [3] <https://docs.djangoproject.com/en/2.0/ref/contrib/admin/> [dostęp: 10 lutego 2018]
- [4] <http://www.django-rest-framework.org/> [dostęp: 10 lutego 2018]
- [5] <https://docs.djangoproject.com/en/2.0/topics/i18n/translation/> [dostęp: 10 lutego 2018]
- [6] <https://docs.djangoproject.com/en/2.0/topics/db/models/> [dostęp: 10 lutego 2018]
- [7] <https://docs.djangoproject.com/pl/2.0/intro/overview/> [dostęp: 10 lutego 2018]
- [8] A. Mele, "Django. Praktyczne tworzenie aplikacji sieciowych" Helion, 2015.
- [9] E. Nemeth, G. Snyder, T. Hein, B. Whaley, "Unix i Linux. Przewodnik administratora systemów.", Wydanie IV, Helion, 2011.