P O L I T E C H N I K A Ś W I Ę T O K R Z Y S K A
Wydział Elektrotechniki, Automatyki i Informatyki

Projekt: Programowanie obiektowe (Java)		
Temat: Aplikacja do obsługi biblioteki dla administratora i użytkownika		
Ocena:	Członkowie zespołu: Przemysław Młynarczyk Paweł Karaś	Grupa: **2ID13A**
		Data oddania projektu: **18.06.2023**

1. Ogólny opis projektu.

Stworzyliśmy projekt „Lectorium” który odzwierciedla aplikację biblioteki. W projekcie używaliśmy takie technologie jak: Maven, JavaFX, SQL Lite, JUnit 5.

2. Informacje na temat funkcjonalności projektu

W projekcie mamy możliwość korzystania z aplikacji w dwóch trybach: użytkownika, oraz administratora.

Użytkownik ma następujące możliwości:

- Możliwość zmiany loginu, hasła, avatara. Możliwość usunięcia swojego konta.
- Możliwość przeglądania katalogu książek. Można wejść w konkretną książkę aby zobaczyć informację na jej temat. W tym miejscu można również zarezerwować książkę.
- Możliwość sprawdzania informacji o książkach (w zakładce Kategorie) z danej tematyki.
- Możliwość sprawdzenia nowo dodanych książek.
- Możliwość sprawdzenia naszych wypożyczeń. W tej zakładce znajdziemy również informację na temat zadłużenia w bibliotece.
- Możliwość sprawdzenia naszych rezerwacji danych książek.
- Sprawdzenie ogólnych informacji na temat biblioteki (Twoja biblioteka, Kontakt).

Administrator ma następujące możliwości:

- Możliwość zmiany loginu, hasła, avatara. Możliwość usunięcia swojego konta.
- Możliwość przeglądania katalogu książek. Można wejść w konkretną książkę aby zobaczyć informację na jej temat. W tym miejscu można również zarezerwować książkę.
- Możliwość sprawdzania informacji o książkach (w zakładce Kategorie) z danej tematyki.
- Możliwość sprawdzenia nowo dodanych książek.
- Możliwość zarządzania katalogiem. A w nim sześć opcji do zarządzania(Dodaj pozycje, Dodaj egzemplarz, Zmodyfikuj pozycje, Zmodyfikuj egzemplarz, Usuń pozycje, Usuń egzemplarz).
- Możliwość zarządzania użytkownikami. A w nim siedem opcji do zarządzania(Dodawanie użytkownika, Zmodyfikowanie użytkownika, Usuwanie użytkownika, Zmodyfikowanie uprawnień, Sprawdzanie wypożyczeń, Sprawdzanie rezerwacji, Blokada kont).
- Sprawdzenie ogólnych informacji na temat biblioteki (Twoja biblioteka, Kontakt).

3. Informacje na temat sposobu uruchamiania projektu oraz jego obsługi

Projekt możemy uruchomić w środowisku Intellij Idea, pobierając go z środowiska Git. Gdy pobierzemy projekt, należy zmienić ścieżkę do pliku. Biblioteka javafx jest dołączona w zdalnym repozytorium.

4. informacje na temat stworzonych klas, metod, funkcji

Klasa wypożyczenia:

```
/**
 * Klasa reprezentująca wypożyczenie egzemplarza.
 */
```

```
/**
 * Konstruktor klasy {@code Wypozyczenia}.
 *
 * @param id_egz              ID egzemplarza.
 * @param naz                 Nazwa egzemplarza.
 * @param data_wypo           Data wypożyczenia.
 * @param data_zwr            Data zwrotu.
 * @param nazwa_autor         Nazwa autora egzemplarza.
 * @param ilosc_przedluzen    Ilość przedłużeń wypożyczenia.
 */
```

Klasa ta, zawiera dla każdego parametru gettery i settery. W kodzie programu są one opisane.

Klasa Users:

```
/**
 * Klasa reprezentująca użytkownika.
 */
```

```
**
 * Konstruktor klasy {@code Users}.
 *
 * @param imie_katalog         Imię użytkownika.
 * @param nazwisko_katalog     Nazwisko użytkownika.
 * @param id_katalog           ID użytkownika.
 * @param czy_admin_katalog    Informacja czy użytkownik ma uprawnienia
administratora.
 * @param czy_zablokowany      Informacja czy użytkownik jest zablokowany.
 */
```

Klasa ta, zawiera dla każdego parametru gettery i settery. W kodzie programu są one opisane.

Klasa User:

```
/**
 * Klasa reprezentująca użytkownika.
 */
```

```
/**
 * Pobiera instancję użytkownika.
 *
 * @return Instancja użytkownika.
 */
```

Klasa ta, zawiera dla każdego parametru gettery i settery. W kodzie programu są one opisane.

Klasa Run:

```
/**
 * Klasa run wykorzystana do stworzenia pliku .jar
 */
```

```
/**
 * Funkcja main uruchamiająca kompilację.
 * @param args argumenty początkowe
 */
```

Klasa Rezerwacje:

```
/**
 * Klasa reprezentująca rezerwacje.
 */
```

```
/**
 * Tworzy nową instancję rezerwacji.
 *
 * @param id_egz         Identyfikator egzemplarza.
 * @param naz            Nazwa rezerwowanego elementu.
 * @param nazwa_autor    Nazwa autora.
 * @param data_ko        Data zakończenia rezerwacji.
 * @param data_rez       Data rozpoczęcia rezerwacji.
 * @param prze_rez       Informacja o możliwości przedłużenia rezerwacji.
 * @param anul_rez       Informacja o możliwości anulowania rezerwacji.
 */
```

Klasa ta, zawiera dla każdego parametru gettery i settery. W kodzie programu są one opisane.

Klasa Main:

```
/**
 * Klasa główna aplikacji, dziedzicząca po klasie Application z JavaFX.
 * Zarządza startem aplikacji, inicjalizacją sceny logowania oraz obsługą
przeciągania okna.
 */
```

```
/**
 * Obiekty {@code db_parent}, {@code db_getData}, {@code db_setData},
{@code db_updateData} i {@code db_deleteData}
 * reprezentują instancje klas do obsługi bazy danych.
 * Każdy obiekt odpowiada za różne operacje na bazie danych, takie jak
pobieranie danych, zapisywanie, aktualizowanie i usuwanie.
 */
```

```
/**
 * Metoda startująca aplikację, odpowiedzialna za inicjalizację sceny
logowania oraz obsługę przeciągania okna.
 *
 * @param stage Główna scena aplikacji.
 */
```

Klasa katalog:

```
/**
 * Klasa reprezentująca katalog w systemie.
 */
```

```
/**
 * Konstruktor inicjalizujący katalog na podstawie przekazanych argumentów.
 *
 * @param id_katalog          Identyfikator katalogu.
 * @param nazwa               Nazwa katalogu.
 * @param nazwa_autora        Nazwa autora.
 * @param rok_wydania         Rok wydania.
 * @param wydanie             Wydanie.
 * @param isbn                Numer ISBN.
 * @param jezyk               Język.
 * @param uwagi               Uwagi.
 * @param nazwa_wydawnictwa   Nazwa wydawnictwa.
 * @param nazwa_gatunku       Nazwa gatunku.
 */
```

Klasa ta, zawiera dla każdego parametru gettery i settery. W kodzie programu są one opisane.

Klasa egzemplarze:

```
/**
 * Klasa reprezentująca egzemplarze książek.
 */
```

```
/**
 * Konstruktor inicjalizujący egzemplarze na podstawie przekazanych
argumentów.
 *
 * @param nazwa               Nazwa egzemplarza.
 * @param id_egzemplarze      Id egzemplarza.
 * @param lokalizacja         Lokalizacja egzemplarza.
 * @param czy_dostepne        Czy egzemplarz dostępny.
 * @param data_zwrotu         Data zwrotu egzemplarza.

 */
```

Klasa ta, zawiera dla każdego parametru gettery i settery. W kodzie programu są one opisane.

Klasa DbUpdateData:

```
/**
 * Klasa DbUpdateData to klasa dziedziczącej po klasie DbParent.
 * Posiada ona metody, które usuwają dane.
 *
 * @see org.example.db.DbParent
 *
 */
```

```
/**
 * Metoda loginUpdate do aktualizacji danych logowania dla danego
użytkownika.
 *
```

```
 * @param new_login nowy login
 * @param id        identyfikator użytkownika
 * @param login     aktualny login
 * @return true, jeśli aktualizacja powiodła się; false w przeciwnym
przypadku
 */
```

```
/**
 * Metoda updatePassword aktualizująca hasło użytkownika.
 *
 * @param new_password nowe hasło
 * @param id           ID użytkownika
 * @param password     aktualne hasło
 * @return true, jeśli hasło zostało zaktualizowane, false w przeciwnym
przypadku
 */
```

```
/**
 * Metoda changeAvatar zmieniająca avatar użytkownika.
 *
 * @param imageData ciąg bitowy nowego obrazu
 * @param id ID użytkownika
 * @throws IOException w przypadku problemów z zapisem danych obrazu
 */
```

```
/**
 * Metoda modifyCopy zmienia dane danego egzemplarza.
 *
 * @param czyDostepne czy książka jest dostępna (parametr wykorzystywany
przy wyświetlaniu stanu książki)
 * @param lokalizacja lokalizacja, gdzie dana książka ma swoje miejsce w
bibliotece
 * @param katalog nazwa pozycji
 * @param idEgzemplarze id egzemplarza danej pozycji
 * @return zwraca ilość zmodyfikowanych rekordów
 */
```

```
/**
 * Metoda modifyBook pozwala na modyfikację danych, dla książki o danym id
katalogu.
 *
 * @param rokWydania rok wydania
 * @param wydanie wydanie
 * @param isbn kod ISBN
 * @param jezyk język w którym książka jest napisana
 * @param uwagi uwagi
 * @param idKatalog id danej książki w katalogu
 * @return zwraca ilość zmodyfikowanych rekordów
 */
```

```
/**
 * Metoda changeUser modyfikuje dane użytkownika, o podanym id_uzytkownicy.
 *
 * @param imie imie użytkownika
 * @param nazwisko nazwisko użytkownika
 * @param login login użytkownika
 * @param haslo hasło użytkownika
```

```
 * @param id_uzytkownicy id użytkownika
 * @return zwraca ilość zmodyfikowanych rekordów
 */
```

```
/**
 * Metoda modifyUprawnienia zmienia użytkownika na admina i odwrotnie.
Dzięki tej metodzie admin może zostać zwykłym
 * użytkownikiem i korzystać w zwykły sposób z biblioteki.
 *
 * @param czy_admin pole w bazie przyjmuje wartości "T" i "N". Inne są
odrzucane.
 * @param id_uzytkownicy id użytkownika
 * @return ilość zmodyfikowanych rekordów
 */
```

```
/**
 * Metoda blockUser zmienia status użytkownika dając bądź odbierając
możliwość zalogowania się użytkownika do programu.
 *
 * @param id_user id użytkownika
 * @param block parametr "T" oznacza ustawienie blokady, "N" zdjęcie jej.
 * @return zwraca ilość zmodyfikowanych rekordów w bazie (ilość
użytkowników)
 */
```

```
/**
 * Metoda updateReservation przedluza rezerwacje danego egzemplarza o 7
dni.
 *
 * @param id_egzemplarz id egzemplarza danej książki
 * @return ilość rekordów w bazie, która uległa zmianie
 */
```

```
/**
 * Metoda updateRentalDate przedłuża wypożyczenie danego egzemplarza o 30
dni.
 * Aby metoda przez przypadek nie przedłużyła przypadkowych rekordów w
przypadku gdyby użytkownik
 * wypożyczył ten sam egzemplarz pare razy, podajemy datę końca danego
wypożyczenia.
 *
 * @param id_egzemplarz id egzemplarza danej książki
 * @param id_uzytkownik id użytkownika
 * @param rentalDate data końca wypożyczenia
 * @return true, jeśli przedłużenie się powiodło, false w przeciwnym
przypadku
 */
```

Klasa DbSetData:

```
/**
 * Klasa DbSetData to klasa dziedziczącej po klasie DbParent.
 * Posiada ona metody, które dodają dane do bazy.
 *
 * @see org.example.db.DbParent
 *
 */
```

```
/**
 * Metoda addUser dodaje nowego użytkownika do bazy. Dodawany tak
użytkownik nie musi spełniać żadnych warunków,
 * w przeciwieństwie do funkcji tryRegister, gdzie tam sprawdzanych jest
kilka warunków.
 *
 * @param imie imie użytkownika
 * @param nazwisko nazwisko użytkownika
 * @param login login użytkownika
 * @param haslo hasło użytkownika
 * @param czy_admin pole w bazie przyjmuje wartości "T" i "N". Inne są
odrzucane.
 * @return zwraca ilość dodanych rekordów do bazy
 */
```

```
/**
 * Metoda addAuthor dodająca autora do tabeli autor.
 *
 * @param imie imie autora
 * @param nazwisko nazwisko autora
 * @return true jeżeli udało się dodać autora do bazy, false jeżeli nie
udało się dodać rekord
 */
```

```
/**
 * Metoda addGenre dodająca gatunek do tabeli gatunków.
 *
 * @param nazwa_gatunku
 * @return true jeżeli udało się dodać gatunek do bazy, false jeżeli się
nie udało
 */
```

```
/**
 * Metoda addPublisher dodająca wydawnictwo do tabeli wydawnictw.
 *
 * @param nazwa_wydawnictwa nazwa wydawnictwa
 * @return true jeżeli udało się dodać wydawnictwo do bazy, false jeżeli
się nie udało się
 */
```

```
/**
 * Metoda setRent do stworzenia nowego wypożyczenia egzemplarza książki
przez danego użytkownika.
 *
 * @param egz numer egzemplarza
 * @param id  identyfikator użytkownika
 */
```

```
/**
 * Funkcja setNewCopy dodaje nowy egzemplarz do bazy danych.
 *
 * @param czy_dostepne  status dostępności egzemplarza
 * @param lokalizacja   lokalizacja egzemplarza
 * @param katalog       nazwa katalogu, do którego należy egzemplarz
 * @return liczba wierszy, które zostały dodane do bazy danych
 */
```

```
/**
 * Metoda add_one_record_catalog pozwala dodać nie tylko nową książkę, ale
też nowego autora, wydawnictwo i gatunek, który będzie mógł być
 * potem stosowany dla innych książek.
 * Najpierw przeprowadzamy dodanie książki do katalogu. Jeżeli się to
powiedzie to zwracamy tablice [false,false,false].
 * W przeciwnym wypadku sprawdzamy kolejno czy istnieje dany autor,
wydawnictwo i gatunek. Jeżeli nie istnieje to w odpowiednim
 * polu tabeli dajemy wartość true.
 * Następnie zwracamy tą tabelę do programu.
 *
 * @param  nazwa  nazwa ksiazki
 * @param rok_wydania rok wydania
 * @param wydanie wydanie
 * @param isbn kod ISBN
 * @param jezyk język książki
 * @param uwagi uwagi odnośnie książki
 * @param imie_autora   imie autora
 * @param nazwisko_autora nazwisko autora
 * @param nazwa_gatunku nazwa gatunku książki
 * @param nazwa_wydawnictwa nazwa wydawnictwa wydającego książkę
 * @return zwraca tabelę 3 elementową, typu boolean; pierwszy element
oznacza czy nie istnieje taki autor, drugi - czy nie istnieje taki gatunek,
 * trzeci - czy nie istnieje takie wydawnictwo
 */
```

Klasa DbParent:

```
/**
 *
 * Klasa DbParent jest klasą nadrzędną dla klas zawierających funkcje,
które nawiązują połączenie
 * z bazą danych i dokonują na niej jakichś operacji.
 *
 */
```

```
/**
 * Metoda connectToDatabase nawiązująca połączenie z bazą danych.
 */
```

```
/**
 * Metoda closeConnection zamykająca połączenie z bazą danych.
 */
```

Klasa DbGetData:

```
/**
 * Klasa DbGetData to klasa dziedziczącej po klasie DbParent.
 * Posiada ona metody, które pobierają dane z bazy danych.
 *
 * @see org.example.db.DbParent
 *
 */
```

```
/**
 * Metoda getBooks pobiera informacje o wszystkich książkach, będących w
tabeli "katalog" bazy danych.
 * Dane o każdej książce zapisuje w postaci tablicy Stringów do kontenera
 *
 * @see #books
 */
```

```
/**
 * Metoda getCover pobiera okładkę dla danej książki na podstawie jej ID z
tabeli katalog.
 * Jeżeli w bazie okładki nie ma, zastępuje ją odpowiednim obrazem.
 * @param id ID książki
 *
 */
```

```
/**
 * Metoda getTop do pobrania danych 8 najnowszych książek.
 * Dane są pobierane tylko raz i przechowywane w kontenerze top.
 * Przy następnych wywołaniach, funkcja nic nie robi.
 *
 * @see #top
 */
```

```
/**
 * Metoda getCategories do pobrania listy kategorii i ilości książek w nich
zawartych.
 * Dane są pobierane tylko raz i przechowywane w kontenerze categories.
 * Przy następnych wywołaniach, funkcja nic nie robi.
 *
 * @see #categories
 */
```

```
/**
 * Metoda getHireInformation pobierająca informacje o wszystkich
wypożyczeniach użytkownika o podanym ID.
 * Dane są zapisywane do kontenera rental.
 *
 * @param id ID użytkownika
 * @see #rental
 */
```

```
/**
 * Metoda getRentLimit zwracająca ilość rezerwacji dla użytkownika o
podanym ID.
 * Metoda getRentLimit wykorzystywana jest dla ustalenia czy użytkownik
przekroczył limit rezerwacji.
 *
 * @param id ID użytkownika
 * @return liczba rezerwacji egzemplarzy
 */
```

```
/**
 * Metoda checkHireInformation sprawdzająca informacje o aktualnych
wypożyczeniach użytkownika o podanym ID.
 * Dane zapisywane są do kontenera rental.
 *
 * @param id ID użytkownika
 * @see #rental
 */
```

```
/**
 * Metoda getCopies która pobiera dane odnośnie egzemplarzy danej książki i
umieszcza je w kontenerze copies.
 * Metoda składa się z głównego zapytania query, które pobiera wszystkie
rekordy odnośnie egzemplarzy danej
 * książki, jakie się znajdują w bazie.
 * Następnie, bazując na pobranej danej "czy_dostepne", funkcja w
zależności od parametru sprawdza warunki.
 *
 * Jeżeli "czy_dostępne" = "T", to takie pozostaje, jeżeli "N" to
sprawdzane jest kolejno czy została ona wypożyczona,
 * bądź zarezerwowana. W każdej możliwości sprawdzane jest czy zrobił to
uzytkownik o danym idUzytkownika, czy inny.
 * Jeżeli żadna z możliwości nie jest spełniona, parametr pozostaje równy
"N".
 *
 * @param idKatalog      identyfikator katalogu
 * @param idUzytkownika  identyfikator użytkownika
 * @see #copies
 */
```

```
/**
 * Funkcja printUsers wypisuje do kontenera users dane wszystkich
użytkowników.
 *
 * @see #users
 */
```

```
/**
 * Metoda getReservationInformation pobierająca informacje o rezerwacjach
użytkownika o podanym ID.
 * Dane umieszczane są w kontenerze rental.
 *
 * @param id ID użytkownika
 * @see #rental
 */
```

Klasa DbDeleteData:

```
/**
 * Klasa DbDeleteData to klasa dziedziczącej po klasie DbParent.
 * Posiada ona metody, które usuwają dane.
 *
 * @see org.example.db.DbParent
 *
 */
```

```
/**
 * Metoda deleteCopyFromDatabase usuwa wszystkie rekordy z tabeli
egzemplarze o danej nazwie książki i id egzemplarza
 *
 * @param katalog nazwa książki
 * @param id_egzemplarz id_egzemplarza
 * @return
 */
```

```
/**
 * Metoda deleteBookFromDatabase usuwa książkę o danych parametrach z bazy
danych. Baza danych sama usunie egzemplarze dla danej książki.
 *
 * @param nazwa nazwa książki
 * @param isbn unikalny kod ISBN
 * @param nazwaGatunku nazwa gatunku
 * @param nazwaWydawnictwa nazwa wydawnictwa
 * @return zwróci ilość usuniętych rekordów (książek)
 */
```

```
/**
 * Metoda deleteUser pozwala na usunięcie użytkownika. Baza danych
automatycznie usunie jego wszystkie wypożyczenia i rezerwacje.
 *
 * @param id_user id użytkownika
 * @return zwraca ilość usuniętych rekordów (użytkowników)
 */
```

```
/**
 * Metoda deleteProfile usuwająca profil użytkownika.
 *
 * @param password hasło użytkownika
 * @param id       ID użytkownika
 * @return true, jeśli profil został usunięty, false w przeciwnym przypadku
 */
```

```
/**
 * Metoda deleteReservation usuwa rezerwację danej książki dla danego
użytkownika.
 * Możliwa jest tylko jedna rezerwacja danego egzemplarza w danym czasie,
więc nie tworzymy dodatkowych mechanizmów
 * sprawdzających poprawność wypożyczenia.
 *
 * @param id_uzytkownika - id użytkownika dokonującego wypożyczenia
 * @param id_egzemplarz - id egzemplarza danej książki
 * @return zwraca ilość rekordów usuniętych przez zapytanie
 */
```

Klasa DbAuth:

```
/**
 * Klasa DbAuth to klasa dziedziczącej po klasie DbParent.
 * Posiada ona metody dotyczące logowania i rejestrowania się użytkowników.
 *
 * @see org.example.db.DbParent
 *
 */
```

```
/**
 * Metoda testRegister sprawdzająca dostępność podanego loginu podczas
rejestracji.
 *
 * @param login login do sprawdzenia
 * @return true, jeśli login jest dostępny; false w przeciwnym razie
 */
```

```
/**
 * Metoda tryLogin próbująca zalogować użytkownika.
 *
 * @param login    login użytkownika
 * @param password hasło użytkownika
 * @return true, jeśli logowanie powiodło się; false w przeciwnym razie
 */
```

```
/**
 * Metoda tryRegister próbująca zarejestrować nowego użytkownika.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 * @param login    login użytkownika
 * @param password hasło użytkownika
 * @return true, jeśli rejestracja powiodła się; false w przeciwnym razie
 */
```

```
/**
 * Metoda getImage pobiera i ustawia avatar dla użytkownika na podstawie
jego ID.
 * Następnie metoda przerabia zdjęcie z formy bitowej na obiekt klasy Image
i umieszcza je w instancji User.
 *
 * @param id ID użytkownika
 *
 */
```

Klasa PasswordSkin:

```
/**
 * Klasa reprezentująca skórkę dla pola tekstowego z hasłem.
 * Ta klasa rozszerza klasę `TextFieldSkin` i dostarcza funkcjonalność
 * maskowania tekstu wprowadzanego w polu tekstowym z hasłem.
 */
```

```
/**
 * Konstruktor klasy `PasswordSkin`.
 *
```

```
 * @param textField Pole tekstowe, dla którego tworzona jest skórka.
 */
```

```
/**
 * Metoda maskText maskująca tekst wprowadzany w polu tekstowym z hasłem.
 * Jeśli skórka jest przypisana do pola typu `PasswordField`,
 * zamienia każdy znak tekstu na znak maskujący '•'.
 *
 * @param txt Tekst do zmaskowania.
 * @return Zmaskowany tekst.
 */
```

Klasa appParent:

```
/**
 * Klasa appParent reprezentuje kontroler głównego okna aplikacji.
 * Zawiera metody obsługujące zdarzenia interakcji użytkownika oraz
 * inicjalizujące interfejs użytkownika.
 */
```

```
/**
 * Metoda logout_perform obsługuje zdarzenie wylogowania użytkownika.
 *
 * @param event zdarzenie myszy
 */
```

```
/**
 * Metoda katalog_clicked obsługuje zdarzenie kliknięcia na przycisk
katalogu.
 *
 * @param event zdarzenie myszy
 */
```

```
/**
 * Metoda katalog_clicked obsługuje zdarzenie kliknięcia na przycisk
katalogu z zapytaniem wziętym z pola wyszukiwania.
 *
 * @param event  zdarzenie myszy
 * @param query  zapytanie z wyszukiwania
 */
```

```
/**
 * Metoda kategorie_clicked obsługuje zdarzenie kliknięcia na przycisk
kategorii.
 *
 * @param event zdarzenie myszy
 */
```

```
/**
 * Metoda contact_clicked obsługuje zdarzenie kliknięcia na przycisk
kontaktu.
 *
 * @param event zdarzenie myszy
 */
```

```
/**
 * Metoda glowna_clicked obsługuje zdarzenie kliknięcia na przycisk strony
głównej.
```

```
 *
 * @param event zdarzenie myszy
 */
```

```
/**
 * Metoda library_clicked obsługuje zdarzenie kliknięcia na przycisk
biblioteki.
 *
 * @param event zdarzenie myszy
 */
```

```
/**
 * Metoda yourProfileController_clicked obsługuje zdarzenie kliknięcia na
przycisk profilu użytkownika.
 *
 * @param event zdarzenie myszy
 */
```

```
/**
 * Metoda nowościController_clicked obsługuje zdarzenie kliknięcia na
przycisk nowości.
 *
 * @param event zdarzenie myszy
 */
```

```
/**
 * Metoda font wykorzystuje aktualny interfejs użytkownika i ustawia
czcionki.
 *
 * @param scene obiekt reprezentujący aktualnie przetwarzaną scenę
aplikacji
 */
```

```
/**
 * Metoda katalog_item obsługuje zdarzenie kliknięcia na wiersz danej
książki w katalogu, bądź inne pole.
 *
 * @param event zdarzenie myszy
 * @param id_egz numer id ksiazki w katalogu
 * @param if_admin boolean sprawdzający czy wchodzi admin czy zwykły
użytkownik
 */
```

```
/**
 * Metoda yourHire_clicked obsługuje zdarzenie kliknięcia na pole twoje
wypożyczenia.
 *
 * @param event zdarzenie myszy
 * */
```

```
/**
 * Metoda yourReservation_clicked obsługuje zdarzenie kliknięcia na pole
twoje rezerwacje.
 *
 * @param event zdarzenie myszy
 */
```

```
/**
 * Metoda ovverideLabels dostosowuje wygląd programu i treść bocznego menu,
 * w zależności od tego czy zalogowany jest student, czy administrator.
 * @param scene obiekt reprezentujący aktualnie przetwarzaną scenę
aplikacji
 */
```

```
/**
 * Metoda size_guard ustala minimalne wymiary okna.
 *
 * @param stage referencja do obiektu Stage
 */
```

```
/**
 * Node createPriorityGraphic tworzy grafikę plusa dla pola w tabeli.
 *
 * @return obiekt Node reprezentujący grafikę plusa
 */
```

```
/**
 * Node createConfirmGraphic tworzy grafikę ticka dla pola w tabeli.
 *
 * @return obiekt Node reprezentujący grafikę tick (fajki)
 */
```

```
/**
 * Tworzy createDeleteGraphic grafikę minusa dla pola w tabeli.
 *
 * @return obiekt Node reprezentujący grafikę minusa
 */
```

Klasa catalogManagerController:

```
/**
 * Klasa kontrolera zarządzającego katalogiem.
 */
```

```
/**
 * Metoda init inicjalizuje kontroler.
 * @param imie imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda Katalog_lista tworzy tabelę z listą książek w określonym panelu.
 * @param anchortable panel, w którym ma zostać wyświetlona tabela
 * @param searchbar pole tekstowe do filtrowania listy książek
 */
```

```
/**
 * Metoda font ustawia styl czcionki dla elementów unikalnych dla tej
sceny.
 * @param scene scena, dla której ma zostać ustawiony styl czcionki
 */
```

```
/**
 * Metoda avatar_view ustawia sposób wyświetlania się awatara użytkownika.
 */
```

```
/**
 * Metoda search_init inicjuje wyszukiwanie na podstawie podanego
zapytania.
 * @param event zdarzenie kliknięcia przycisku
 */
```

```
/**
 * Metoda dodaj_egz_button obsługuje dodawanie nowego egzemplarza książki.
Metoda ustawia nazwy pól.
 * Przy kliknięciu w guzik wywołuje funkcje z bazy i zwraca odpowiedni
wynik.
 */
```

```
/**
 * Metoda usun_egz_button obsługuje usunięcie egzemplarza książki. Metoda
ustawia nazwy pól.
 * Przy kliknięciu w guzik wywołuje funkcje z bazy i zwraca odpowiedni
wynik.
 */
```

```
/**
 * Metoda usun_pozycje_button obsługuje usunięcie książki z katalogu.
Metoda ustawia nazwy pól.
 * Przy kliknięciu w guzik wywołuje funkcje z bazy i zwraca odpowiedni
wynik.
 */
```

```
/**
 * Metoda zmodyfikuj_egzemplarze obsługuje modyfikacje egzemplarza książki.
Metoda ustawia nazwy pól.
 * Przy kliknięciu w guzik wywołuje funkcje z bazy i zwraca odpowiedni
wynik.
 */
```

```
/**
 * Metoda add_position obsługuje dodanie książki do katalogu. Metoda
ustawia nazwy pól.
 * Przy kliknięciu w guzik wywołuje funkcje z bazy i zwraca odpowiedni
wynik.
 * Wynik jest zwracany w postaci 3-elementowej tablicy typu boolean.
```

```
 * Odpowiadają one odpowiednio istnienie danego rekordu dla tabel:
Autor,Gatunek,Wydawnictwo.
 * Wartość true oznacza, że wprowadzony rodzaj elementu nie istnieje w
bazie, false - przeciwnie.
 * Następnie ustawiamy pole sec na true i wywołujemy komunikat.
 * Jeżeli użytkownik kliknie znowu w guzik, wchodzimy w drugi obieg tej
samej funkcji i dodajemy
 * nieistniejące dotychczas parametry do docelowych tabel.
 * Na sam koniec przeprowadzamy dodanie książki do katalogu.
 */
```

```
/**
 * Metoda zmodyfikuj_pozycje obsługuje modyfikacje książki z katalogu.
Metoda ustawia nazwy pól.
 * Przy kliknięciu w guzik wywołuje funkcje z bazy i zwraca odpowiedni
wynik.
 */
```

```
/**
 * Metoda hide_pane wywoływana jest po kliknięciu guzika zamknięcia panelu
pop-up.
 * Przy kliknięciu w guzik chowane są wszystkie przymioty funkcji
 */
```

Klasa yourReservationController:

```
/**
 * Klasa {@code yourReservationsController} jest kontrolerem widoku
informacji o rezerwacjach aktualnie zalogowanego użytkownika.
 * Odpowiada za obsługę interakcji użytkownika, wyświetlanie informacji,
udostępnia funkcje przedłużania i anulowania rezerwacji oraz inicjalizację
widoku.
 * Dziedziczy po klasie {@link appParent}, aby działać w kontekście
głównego okna aplikacji.
 */
```

```
/**
 * Metoda {@code init} inicjalizuje widok ekranu domowego.
 * Ustawia tekst w polu nametag, wczytuje obrazek awatara użytkownika oraz
wywołuje metody odpowiedzialne za wyświetlanie obrazków i ustawienie stylu
labelglowna.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda Lista_Hire wyświetlająca listę wypożyczeń dla danego
identyfikatora.
 *
 * @param id Identyfikator wypożyczenia
 */
```

```
/**
 * Metoda {@code avatar_view} ustawia clipping dla obrazka avatara, aby
uzyskać efekt zaokrąglonych rogów.
 */
```

```
/**
 * Metoda font inicjalizująca styl czcionki dla elementów w scenie.
 * Wywołuje również metodę inicjalizującą styl czcionki dla klasy
nadrzędnej.
 *
 * @param scene obiekt {@link Scene} reprezentujący scenę JavaFX
 * @see appParent#font(Scene)
 */
```

Klasa yourRentsController:

```
/**
 * Klasa {@code yourRentsController} jest kontrolerem widoku informacji o
wypożyczeniach aktualnie zalogowanego użytkownika.
 * Odpowiada za obsługę interakcji użytkownika, wyświetlanie informacji,
udostępnia funkcje przedłużania wypożyczenia oraz inicjalizację widoku.
 * Dziedziczy po klasie {@link appParent}, aby działać w kontekście
głównego okna aplikacji.
 */
```

```
/**
 * Metoda {@code init} inicjalizuje widok ekranu domowego.
 * Ustawia tekst w polu nametag, wczytuje obrazek awatara użytkownika oraz
wywołuje metody odpowiedzialne za wyświetlanie obrazków i ustawienie stylu
labelglowna.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda Lista_Hire wyświetlająca listę wypożyczeń dla danego użytkownika
na podstawie podanego identyfikatora.
 *
 * @param id Identyfikator użytkownika
 */
```

```
/**
 * Metoda {@code avatar_view} ustawia clipping dla obrazka avatara, aby
uzyskać efekt zaokrąglonych rogów.
 */
```

```
/**
 * Metoda font inicjalizująca styl czcionki dla elementów w scenie.
 * Wywołuje również metodę inicjalizującą styl czcionki dla klasy
nadrzędnej.
 *
 * @param scene obiekt {@link Scene} reprezentujący scenę JavaFX
 * @see appParent#font(Scene)
 */
```

```
/**
 * Metoda check_hire_information obsługująca sprawdzanie informacji o
wypożyczeniach po kliknięciu przycisku.
 *
 * @param event Obiekt reprezentujący zdarzenie kliknięcia myszą
 */
```

```
/**
 * Metoda actual_button obsługująca zmianę wyglądu przycisków na podstawie
typu.
 *
 * @param event Obiekt reprezentujący zdarzenie kliknięcia myszą
 * @param type  Typ przycisku (true - "Aktualne", false - "Wszystkie")
 */
```

Klasa yourProfileController:

```
/**
 * Klasa {@code yourProfileController} jest kontrolerem widoku informacji o
profilu aktualnie zalogowanego użytkownika.
 * Odpowiada za obsługę interakcji użytkownika, wyświetlanie informacji o
jego profilu, udostępnia funkcje modyfikacji
 * danych profilu oraz inicjalizację widoku.
 * Dziedziczy po klasie {@link appParent}, aby działać w kontekście
głównego okna aplikacji.
 */
```

```
/**
 * Metoda {@code avatar_view} ustawia clipping dla obrazka avatara, aby
uzyskać efekt zaokrąglonych rogów.
 */
```

```
/**
 * Metoda {@code avatar_view_profile} ustawia clipping dla większego
obrazka avatara, aby uzyskać efekt zaokrąglonych rogów.
 */
```

```
/**
 * Metoda font inicjalizująca styl czcionki dla elementów w scenie.
 * Wywołuje również metodę inicjalizującą styl czcionki dla klasy
nadrzędnej.
 *
 * @param scene obiekt {@link Scene} reprezentujący scenę JavaFX
 * @see appParent#font(Scene)
 */
```

```
/**
 * Metoda {@code init} inicjalizuje widok ekranu domowego.
 * Ustawia tekst w polu nametag, wczytuje obrazek awatara użytkownika oraz
wywołuje metody odpowiedzialne za wyświetlanie obrazków i ustawienie stylu
labelglowna.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda {@code search_init} obsługuje inicjalizację wyszukiwania po
wciśnięciu przycisku lub klawisza Enter w polu searchbar.
 * Pobiera zapytanie z pola searchbar i przekazuje je do metody
katalog_clicked w celu obsługi wyszukiwania.
 *
 * @param event zdarzenie kliknięcia myszy
 */
```

```
/**
 * Metoda haslo_c obsługuje zdarzenie kliknięcia przycisku "Hasło".
 * Wywołuje metodę `wash_effects()` i ustawia widoczność pola "password" na
true.
 *
 * @param event obiekt reprezentujący zdarzenie myszy
 */
```

```
/**
 * Metoda login_c obsługuje zdarzenie kliknięcia przycisku "Login".
 * Wywołuje metodę `wash_effects()` i ustawia widoczność pola "login" na
true.
 *
 * @param event obiekt reprezentujący zdarzenie myszy
 */
```

```
/**
 * Metoda login_change obsługuje zdarzenie kliknięcia przycisku "Zmień
login".
 * Wywołuje metodę `wash_effects()` i wykonuje operacje zmiany loginu w
bazie danych.
 * Wyświetla odpowiednie komunikaty na podstawie rezultatu operacji.
 *
 * @param event obiekt reprezentujący zdarzenie myszy
 */
```

```
/**
 * Metoda password_change obsługuje zdarzenie kliknięcia przycisku "Zmień
hasło".
 * Wywołuje metodę `wash_effects()` i wykonuje operacje zmiany hasła w
bazie danych.
 * Wyświetla odpowiednie komunikaty na podstawie rezultatu operacji.
 *
 * @param event obiekt reprezentujący zdarzenie myszy
 */
```

```
/**
 * Metoda profile_d obsługuje zdarzenie kliknięcia przycisku "Profil".
 * Wywołuje metodę `wash_effects()` i ustawia widoczność pola "usun" na
true.
 * Ustawia styl ramki przycisku "profile_delete".
 *
 * @param event obiekt reprezentujący zdarzenie myszy
 */
```

```
/**
 * Metoda delete_profile obsługuje zdarzenie kliknięcia przycisku "Usuń
profil".
 * Wywołuje metodę `wash_effects()` i wykonuje operacje usunięcia profilu z
bazy danych.
 * Wyświetla odpowiednie komunikaty na podstawie rezultatu operacji.
 *
 * @param event obiekt reprezentujący zdarzenie myszy
 */
```

```
/**
 * Metoda wash_effects resetuje efekty wizualne ustawione na elementach
interfejsu.
 * Ukrywa pola "password", "usun" i "login".
 */
```

```
/**
 * Metoda avatar_swap obsługuje zdarzenie zamiany avatara.
 * Otwiera okno dialogowe umożliwiające wybór pliku z obrazem.
 * Po wybraniu pliku, przekształca go na obiekt Image, zmieniając aktualny
avatar użytkownika.
 * Następnie przekształca obraz na format BufferedImage, a następnie na
tablicę bajtów.
 * Wywołuje operację zmiany avatara w bazie danych, a następnie ponownie
inicjuje zdarzenie dla elementu "avatar".
 * W przypadku wystąpienia błędu podczas operacji, wyświetla odpowiedni
komunikat.
 *
 * @param event obiekt reprezentujący zdarzenie myszy
 */
```

Klasa yourLibraryController:

```
/**
 * Klasa {@code yourLibraryController} jest kontrolerem widoku informacji o
bibliotece.
 * Odpowiada za obsługę interakcji użytkownika, wyświetlanie informacji o
bibliotece oraz inicjalizację widoku.
 * Dziedziczy po klasie {@link appParent}, aby działać w kontekście
głównego okna aplikacji.
 */
```

```
/**
 * Metoda {@code init} inicjalizuje widok ekranu domowego.
 * Ustawia tekst w polu nametag, wczytuje obrazek awatara użytkownika oraz
wywołuje metody odpowiedzialne za wyświetlanie obrazków i ustawienie stylu
labelglowna.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda {@code avatar_view} ustawia clipping dla obrazka avatara, aby
uzyskać efekt zaokrąglonych rogów.
 */
```

```
/**
 * Metoda {@code images_view} ustawia clipping dla obrazków
 */
```

```
/**
 * Metoda font inicjalizuje styl czcionki dla elementów w scenie.
 * Wywołuje również metodę inicjalizującą styl czcionki dla klasy
nadrzędnej.
 *
 * @param scene obiekt {@link Scene} reprezentujący scenę JavaFX
 * @see appParent#font(Scene)
 */
```

```
/**
 * Metoda {@code search_init} obsługuje inicjalizację wyszukiwania po
wciśnięciu przycisku lub klawisza Enter w polu searchbar.
 * Pobiera zapytanie z pola searchbar i przekazuje je do metody
katalog_clicked w celu obsługi wyszukiwania.
 *
 * @param event zdarzenie kliknięcia myszy
 */
```

Klasa lastestBooksController:

```
/**
 * Klasa {@code latestBooksController} reprezentuje kontroler widoku
najnowszych książek.
 * Odpowiada za logikę interakcji użytkownika z widokiem, taką jak
wyszukiwanie, wyświetlanie informacji o książkach itp.
 *
 * @see appParent
 */
```

```
/**
 * Metoda {@code search_init} inicjalizuje wyszukiwanie.
 * Pobiera z pola tekstowego zapytanie wyszukiwania i wywołuje metodę
{@code katalog_clicked} z odpowiednimi parametrami.
 *
 * @param event zdarzenie kliknięcia myszą
 */
```

```
/**
 * Metoda font inicjalizuje styl czcionki dla elementów w scenie.
 * Wywołuje również metodę inicjalizującą styl czcionki dla klasy
nadrzędnej.
 *
 * @param scene obiekt {@link Scene} reprezentujący scenę JavaFX
 * @see appParent#font(Scene)
 */
```

```
/**
 * Metoda init inicjalizująca kontroler widoku najnowszych książek.
 * Ustawia informacje o użytkowniku, inicjalizuje dane dotyczące
najnowszych książek,
 * a także wyświetla odpowiednie elementy w interfejsie użytkownika.
```

```
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda setLabelText ustawiająca teksty etykiety  i obraz dla danego
VBoxa.
 *
 * @param vb         obiekt VBox, dla którego ustawiany jest tekst etykiety
 * @param nazwa      nazwa książki
 * @param autor      nazwa autora
 * @param id_katalog identyfikator katalogowy książki
 * @param i          numer indeksu VBoxa
 */
```

```
/**
 * Metoda {@code avatar_view} ustala widok awatara.
 * Tworzy okrągłe przycięcie dla awatara.
 */
```

```
/**
 * Metoda refer_to_book ustawiająca przejście do danej książki po
kliknięciu jej okładki.
 */
```

Klasa homeController:

```
/**
 * Klasa {@code homeController} jest kontrolerem widoku domowego ekranu
aplikacji.
 * Odpowiada za obsługę interakcji użytkownika, wyświetlanie informacji
oraz inicjalizację widoku.
 * Dziedziczy po klasie {@link appParent}, aby działać w kontekście
głównego okna aplikacji.
 */
```

```
/**
 * Metoda {@code init} inicjalizuje widok ekranu domowego.
 * Ustawia tekst w polu nametag, wczytuje obrazek awatara użytkownika oraz
wywołuje metody odpowiedzialne za wyświetlanie obrazków i ustawienie stylu
labelglowna.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda font inicjalizuje styl czcionki dla elementów w scenie.
 * Wywołuje również metodę inicjalizującą styl czcionki dla klasy
nadrzędnej.
 *
 * @param scene obiekt {@link Scene} reprezentujący scenę JavaFX
```

```
 * @see appParent#font(Scene)
 */
```

```
/**
 * Metoda {@code avatar_view} ustawia clipping dla obrazka avatara, aby
uzyskać efekt zaokrąglonych rogów.
 */
```

```
/**
 * Metoda {@code images_view} ustawia clipping dla obrazków
 */
```

```
/**
 * Metoda {@code search_init} obsługuje inicjalizację wyszukiwania po
wciśnięciu przycisku lub klawisza Enter w polu searchbar.
 * Pobiera zapytanie z pola searchbar i przekazuje je do metody
katalog_clicked w celu obsługi wyszukiwania.
 *
 * @param event zdarzenie kliknięcia myszy
 */
```

```
/**
 * Metoda {@code menu_panels} obsługuje kliknięcia na panele menu.
 * W zależności od klikniętego obrazka, wywołuje odpowiednią funkcję.
 *
 * @param event zdarzenie kliknięcia myszy
 */
```

Klasa contactController:

```
/**
 * Klasa {@code contactController} jest kontrolerem widoku kontaktu z
biblioteką.
 * Odpowiada za obsługę interakcji użytkownika, wyświetlanie informacji
oraz inicjalizację widoku.
 * Dziedziczy po klasie {@link appParent}, aby działać w kontekście
głównego okna aplikacji.
 */
```

```
/**
 * Metoda {@code init} inicjalizuje widok ekranu domowego.
 * Ustawia tekst w polu nametag, wczytuje obrazek awatara użytkownika oraz
wywołuje metody odpowiedzialne za wyświetlanie obrazków i ustawienie stylu
labelglowna.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda {@code images_view} ustawia clipping dla obrazków
 */
```

```
/**
 * Metoda {@code avatar_view} ustawia clipping dla obrazka avatara, aby
uzyskać efekt zaokrąglonych rogów.
 */
```

```
/**
 * Metoda font inicjalizuje styl czcionki dla elementów w scenie.
 * Wywołuje również metodę inicjalizującą styl czcionki dla klasy
nadrzędnej.
 *
 * @param scene obiekt {@link Scene} reprezentujący scenę JavaFX
 * @see appParent#font(Scene)
 */
```

```
/**
 * Metoda {@code search_init} obsługuje inicjalizację wyszukiwania po
wciśnięciu przycisku lub klawisza Enter w polu searchbar.
 * Pobiera zapytanie z pola searchbar i przekazuje je do metody
katalog_clicked w celu obsługi wyszukiwania.
 *
 * @param event zdarzenie kliknięcia myszy
 */
```

Klasa categoriesController:

```
/**
 * Klasa {@code categoriesController} reprezentuje kontroler widoku
kategorii.
 * Zarządza interakcjami i logiką związaną z widokiem kategorii.
 * Rozszerza klasę {@link appParent}, aby dziedziczyć pewne funkcjonalności
z klasy nadrzędnej.
 */
```

```
/**
 * Metoda {@code init} inicjalizuje kontroler kategorii.
 * Ustawia imię i nazwisko użytkownika, obrazek awatara, widoki kategorii
oraz styl etykiety kategorii.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda {@code avatar_view} ustala widok awatara.
 * Tworzy okrągłe przycięcie dla awatara.
 */
```

```
/**
 * Metoda {@code search_init} inicjalizuje wyszukiwanie.
 * Pobiera z pola tekstowego zapytanie wyszukiwania i wywołuje metodę
{@code katalog_clicked} z odpowiednimi parametrami.
 *
 * @param event zdarzenie kliknięcia myszą
 */
```

```
/**
 * Metoda {@code categories_init} inicjalizuje kategorie.
 * Pobiera dane kategorii z bazy danych i tworzy widok kategorii na
podstawie tych danych.
 */
```

```
/**
 * Metoda {@code images_view} ustala wygląd pól reklamowych.
 */
```

```
/**
 * Metoda {@code menu_panels} obsługuje kliknięcie na panele reklamowe.
 * W zależności od kliknietego panelu, wywołuje odpowiednie metody.
 *
 * @param event zdarzenie kliknięcia myszą
 */
```

Klasa catalogItemController:

```
/**
 * Klasa {@code catalogItemController} jest kontrolerem widoku informacji o
egzemplarzach danej pozycji z katalogu.
 * Odpowiada za obsługę interakcji użytkownika, wyświetlanie informacji,
udostępnia funkcje rezerwacji oraz inicjalizację widoku.
 * Dziedziczy po klasie {@link appParent}, aby działać w kontekście
głównego okna aplikacji.
 */
```

```
/**
 * Metoda font inicjalizuje styl czcionki dla elementów w scenie.
 * Wywołuje również metodę inicjalizującą styl czcionki dla klasy
nadrzędnej.
 *
 * @param scene obiekt {@link Scene} reprezentujący scenę JavaFX
 * @see appParent#font(Scene)
 */
```

```
/**
 * Metoda load wczytująca dane dla określonego identyfikatora egzemplarza.
 *
 * @param id_egz Identyfikator egzemplarza
 */
```

```
/**
 * Metoda {@code init} inicjalizuje widok ekranu domowego.
 * Ustawia tekst w polu nametag, wczytuje obrazek awatara użytkownika oraz
wywołuje metody odpowiedzialne za wyświetlanie obrazków i ustawienie stylu
labelglowna.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda {@code avatar_view} ustawia clipping dla obrazka avatara, aby
uzyskać efekt zaokrąglonych rogów.
 */
```

```
/**
 * Metoda egzemplarz_lista wyświetlająca listę egzemplarzy dla danej
książki.
 *
 * @param id Identyfikator książki
 */
```

```
/**
 * Metoda {@code cover_view} ustala wygląd okładki
 */
```

```
/**
 * Metoda {@code search_init} obsługuje inicjalizację wyszukiwania po
wciśnięciu przycisku lub klawisza Enter w polu searchbar.
 * Pobiera zapytanie z pola searchbar i przekazuje je do metody
katalog_clicked w celu obsługi wyszukiwania.
 *
 * @param event zdarzenie kliknięcia myszy
 */
```

```
/**
 * Metoda back_to_prv obsługuje zdarzenie kliknięcia przycisku powrotu do
poprzedniego widoku.
 * <p>
 * Jeśli użytkownik jest administratorem, wczytuje plik FXML dla widoku
panelu administratora (admin_home.fxml).
 * Jeśli użytkownik jest zwykłym użytkownikiem, wczytuje plik FXML dla
widoku głównego (home.fxml).
 * Następnie tworzy nową scenę na podstawie wczytanego rodzica i ustawia ją
dla obiektu Stage reprezentującego aktualne okno aplikacji.
 * Metoda show() jest wywoływana na obiekcie Stage w celu wyświetlenia
nowego widoku.
 *
 * @param event zdarzenie kliknięcia myszą na przycisk powrotu
 */
```

Klasa catalogController:

```
/**
 * Klasa {@code catalogController} jest kontrolerem widoku domowego ekranu
aplikacji.
 * Odpowiada za obsługę interakcji użytkownika, wyświetlanie informacji
oraz inicjalizację widoku katalogu książek.
 * Dziedziczy po klasie {@link appParent}, aby działać w kontekście
głównego okna aplikacji.
 */
```

```
/**
 * Metoda Katalog_lista odpowiedzialna za wyświetlanie listy katalogu.
 * Pobiera dane z bazy danych, inicjalizuje tabelę i wyświetla listę
książek.
 * Przyjmuje wszystkie dostępne rekordy z katalogu i wyświetla je w tabeli.
 * Umożliwia filtrowanie i wyszukiwanie po nazwie, autorze, ISBN lub
gatunku.
 * Obsługuje podwójne kliknięcie myszą na rekord, które otwiera szczegóły
książki.
 *
 */
```

```
/**
 * Metoda Katalog_lista odpowiedzialna za wyświetlanie listy katalogu.
 * Pobiera dane z bazy danych, inicjalizuje tabelę i wyświetla listę
książek.
 * Przyjmuje wszystkie dostępne rekordy z katalogu i wyświetla je w tabeli.
 * Umożliwia filtrowanie i wyszukiwanie po nazwie, autorze, ISBN lub
gatunku.
 * Obsługuje podwójne kliknięcie myszą na rekord, które otwiera szczegóły
książki.
 *
 * @param query Opcjonalny parametr - dodaje od razu filtr do przeszukania
katalogu (opcjonalne)
 */
```

```
/**
 * Metoda {@code init} inicjalizuje widok ekranu domowego.
 * Ustawia tekst w polu nametag, wczytuje obrazek awatara użytkownika oraz
wywołuje metody odpowiedzialne za wyświetlanie obrazków i ustawienie stylu
labelglowna.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda avatar_view odpowiedzialna za wyświetlanie awatara użytkownika.
 * Tworzy okrągły wycinek (clip) o określonym promieniu wokół awatara,
 * który jest ustawiany jako wycinek dla elementu avatar.
 */
```

```
/**
 * Metoda search_init inicjalizuje wyszukiwanie w katalogu na podstawie
wprowadzonego zapytania.
 * Pobiera tekst z pola wyszukiwania (searchbar) i wywołuje metodę
katalog_clicked(event, query).
 *
 * @param event Obiekt zdarzenia myszy (MouseEvent)
 */
```

```
/**
 * ScrollBar getVerticalScrollBar pobiera pasek przewijania wertykalnego
dla podanej tabeli (TableView).
 *
 * @param tableView Tabela, dla której ma zostać pobrany pasek przewijania
(TableView)
 * @return Pasek przewijania wertykalnego dla tabeli (ScrollBar) lub null,
jeśli nie znaleziono
 */
```

```
/**
 * Metoda loadNextRecords ładuje kolejne rekordy do listy katalogu.
 * Dodaje do listy katalogu kolejne rekordy z bazy danych,
 * zaczynając od ostatnio wczytanego indeksu.
```

```
 * Określa ilość rekordów do wczytania i zwiększa indeks ostatnio
wczytanego rekordu.
 *
 * @param items Lista obserwowalna, do której mają zostać dodane kolejne
rekordy katalogu (ObservableList)
 */
```

Klasa userManagerController:

```
/**
 * Klasa {@code userManagerController} jest kontrolerem widoku informacji o
egzemplarzach danej pozycji z katalogu.
 * Odpowiada za obsługę interakcji użytkownika, wyświetlanie informacji,
udostępnia funkcje zarządzaniem użytkownikami.
 * Dziedziczy po klasie {@link appParent}, aby działać w kontekście
głównego okna aplikacji.
 */
```

```
/**
 * Metoda {@code init} inicjalizuje widok ekranu domowego.
 * Ustawia tekst w polu nametag, wczytuje obrazek awatara użytkownika oraz
wywołuje metody odpowiedzialne za wyświetlanie obrazków i ustawienie stylu
labelglowna.
 *
 * @param imie     imię użytkownika
 * @param nazwisko nazwisko użytkownika
 */
```

```
/**
 * Metoda font inicjalizuje styl czcionki dla elementów w scenie.
 * Wywołuje również metodę inicjalizującą styl czcionki dla klasy
nadrzędnej.
 *
 * @param scene obiekt {@link Scene} reprezentujący scenę JavaFX
 * @see appParent#font(Scene)
 */
```

```
/**
 * Metoda {@code avatar_view} ustawia clipping dla obrazka avatara, aby
uzyskać efekt zaokrąglonych rogów.
 */
```

```
/**
 * Metoda {@code search_init} obsługuje inicjalizację wyszukiwania po
wciśnięciu przycisku lub klawisza Enter w polu searchbar.
 * Pobiera zapytanie z pola searchbar i przekazuje je do metody
katalog_clicked w celu obsługi wyszukiwania.
 *
 * @param event zdarzenie kliknięcia myszy
 */
```

```
/**
 * Metoda Katalog_lista_adminUser wyświetla listę użytkowników w panelu
administratora.
 * Pobiera dane użytkowników z bazy danych i tworzy tabelę z odpowiednimi
kolumnami.
```

```
 * Każdy wiersz tabeli reprezentuje jednego użytkownika.
 * Możliwe jest dodawanie, modyfikowanie, zmienianie uprawnień i usuwanie
użytkowników.
 * Metoda korzysta z różnych kontrolek, takich jak TextField, Button,
Label, oraz z metod dostępu do bazy danych.
 * Wyświetla również komunikaty o wynikach operacji dodawania, modyfikacji,
zmiany uprawnień i usuwania użytkowników.
 */
```

```
/**
 * Metoda dodaj_uzytkownika obsługuje zdarzenie dodawania nowego
użytkownika.
 * Wyświetla formularz dodawania użytkownika oraz obsługuje logikę
dodawania użytkownika do bazy danych.
 * Wyświetla komunikat o wyniku operacji dodawania użytkownika.
 */
```

```
/**
 * Metoda zmodyfikuj_uzytkownika obsługuje zdarzenie modyfikowania
użytkownika.
 * Wyświetla formularz modyfikacji użytkownika oraz obsługuje logikę
modyfikacji użytkownika w bazie danych.
 * Wyświetla komunikat o wyniku operacji modyfikacji użytkownika.
 */
```

```
/**
 * Metoda zmodyfikuj_uprawnienia_uzytkownika obsługuje zdarzenie zmiany
uprawnień użytkownika.
 * Wyświetla formularz zmiany uprawnień użytkownika oraz obsługuje logikę
zmiany uprawnień w bazie danych.
 * Wyświetla komunikat o wyniku operacji zmiany uprawnień użytkownika.
 */
```

```
/**
 * Metoda usun_uzytkownika obsługuje zdarzenie usuwania użytkownika.
 * Wyświetla formularz usuwania użytkownika oraz obsługuje logikę usuwania
użytkownika z bazy danych.
 * Wyświetla komunikat o wyniku operacji usuwania użytkownika.
 */
```

```
/**
 * Metoda zablokuj_uzytkownika odpowiedzialna za zablokowanie lub
odblokowanie użytkownika.
 * Aktualizuje stan blokady w bazie danych na podstawie podanego loginu
użytkownika.
 * Po wykonaniu operacji wyświetla odpowiedni komunikat.
 */
```

```
/**
 * Metoda rent_activate obsługująca aktywację wypożyczeń użytkownika po
naciśnięciu klawisza ENTER.
 * Sprawdza poprawność wprowadzonego identyfikatora użytkownika i
inicjalizuje listę wypożyczeń.
 * Wyświetla wynik wyszukiwania.
 *
 * @param event Obiekt zdarzenia naciśnięcia klawisza
 */
```

```
/**
 * Metoda check_rent obsługująca sprawdzanie wypożyczeń użytkownika.
 * Ustawia flagę activate_diff na true i wyświetla listę wypożyczeń.
 *
 * @param event Obiekt zdarzenia kliknięcia myszą
 */
```

```
/**
 * Metoda check_res obsługująca sprawdzanie rezerwacji użytkownika.
 * Ustawia flagę activate_diff na false i wyświetla listę rezerwacji.
 *
 * @param event Obiekt zdarzenia kliknięcia myszą
 */
```

```
/**
 * Metoda init_rent_list inicjalizująca listę wypożyczeń dla danego
identyfikatora użytkownika.
 * Pobiera informacje o wypożyczeniach z bazy danych i wyświetla je w
tabeli.
 * Oblicza ewentualną grzywnę za opóźnione zwroty.
 *
 * @param id Identyfikator użytkownika
 */
```

```
/**
 * Metoda init_res_list inicjalizująca listę rezerwacji dla danego
identyfikatora użytkownika.
 * Pobiera informacje o rezerwacjach z bazy danych i wyświetla je w tabeli.
 * Tworzy przyciski do przedłużania i anulowania rezerwacji.
 *
 * @param id Identyfikator użytkownika
 */
```

```
/**
 * Funkcja hide_panes chowa wszystkie elementy dla pop-upów.
 * @param event Obiekt zdarzenia kliknięcia myszą
 */
```

5. Podział pracy

Imie Nazwisko	Podział pracy %
Paweł Karaś	50%
Przemysław Młynarczyk	50%