

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №2**

з дисципліни

**«**Бази даних і засоби управління**»**

Виконав студент ІII курсу

ФПМ групи КВ-83

Глеб В.Ю.

Перевірив: Павловський В.І.

Київ – 2020

**Ознайомлення з базовими операціями СУБД PostgreSQL**

*Завдання* роботи полягає у наступному:

1. Виконати нормалізацію бази даних, яка була створена у лабораторній роботі №1, до третьої нормальної форми (3НФ);
2. Реалізувати функціональні вимоги, наведені нижче.

*Функціональні вимоги*

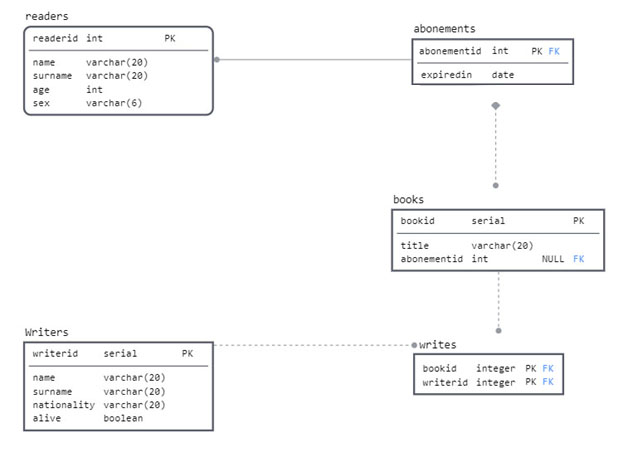
1. Реалізувати внесення, редагування та вилучення даних у базі засобами консольного інтерфейсу;
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі;
3. Забезпечити реалізацію пошуку за двома-трьома атрибутами з двох сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як перелічення, для логічного типу – значення True/False, для дат – у рамках діапазону дат;
4. Забезпечити реалізацію повнотекстового пошуку за будь-яким текстовим атрибутом бази даних засобами PostgreSQL з виділенням знайденого фрагменту.

*Вимоги до інтерфейсу користувача*

1. Використовувати консольний інтерфейс користувача.

**Варіант 3**

**Нормалізована логічна модель даних БД «Бібліотека»**



**Рис 2.1 Нормалізована логічна модель даних БД «Бібліотека»**

Усі таблиці (відношення) знаходяться в 3 НФ тому, що у кожній із них:

1. Всі атрибути є атомарними та відсутні повторення рядків (1НФ);
2. Первинний ключ складається лише з одного атрибуту (2НФ);
3. Кожний не первинний атрибут не є транзитивно залежним від первинного ключа (3НФ).

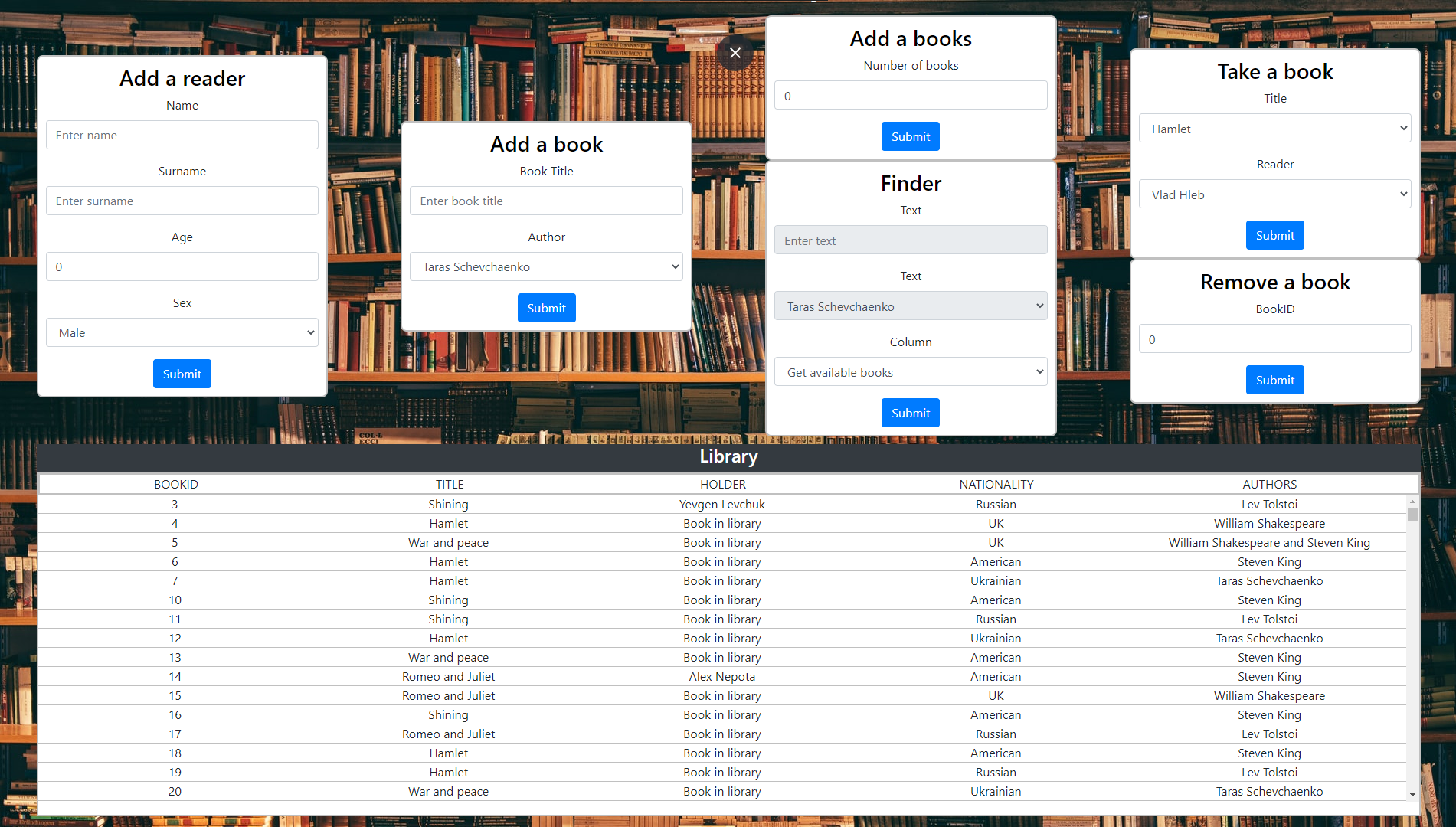
**Опис програми**

Програма створена для управління базою даних за допомогою базових операцій СУБД PostgreSQL та реалізовує функціональні вимоги, що наведені у завданні. Програма складається з 5 модулів:

1. index.js – точка входу до програми, запускає сервер, який приймає запити;
2. router.js – обробляє запити до сервера, підключає файл controller.js;
3. controller.js – обробляє логіку отримання даних. Підключає файл tables.js;
4. tables.js – містить класи, які відповідають за дані в таблицях БД. Для кожної таблиці створений свій клас;

**Структура меню програми**

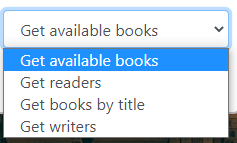
Меню програми

****

**Рис 2.2** **Меню програми**

**Інтерфейс програми**

1. Форма “Add a reader” дозволяє додати читача в бібліотеку. Передає на сервер (POST-запит) дані користувача. Дані зразу з’являються в інтерфейсі;
2. Форма “Add a book” дозволяє добавити книжку в бібліотеку. Передає на сервер (POST-запит) з назвою книжки і іменем автора. Книжка з’являється в вікні “Library”;
3. Форма “Add books” дозволяє заповнити бібліотеку книжками. Передає на сервер (POST-запит) кількість необхідних книжок. Автори і назви беруться рандомно з вже існуючих книжок;
4. Форма “Finder” дозволяє шукати книжки, авторів, читачів(GET-запит):



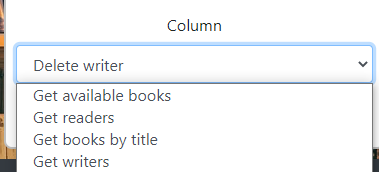
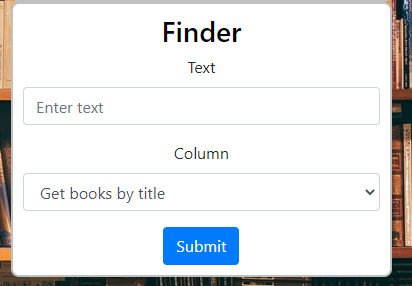
**Рис 2.3 Випадаюче вікно форми Finder**

* 1. “Get available books” виводить всі книжки, які знаходяться в бібліотеці
  2. “Get readers” виводить дані всіх читачів;
  3. “Get books by title” виводить всі книжки з введеною вище назвою;
  4. “Get writers” виводить всіх авторів і інофрмацію про них;

1. Форма “Take a book” дозволяє взяти книгу з бібліотеки. Книга буде записана на читача, який був введений в формі.(PUT-запит);
2. Форма “Remove a book” видаляє книгу з бібліотеки.(DELETE-запит);
3. Таблиця “Library” містить актуальну інформацію про всі книжки і їх читачів.

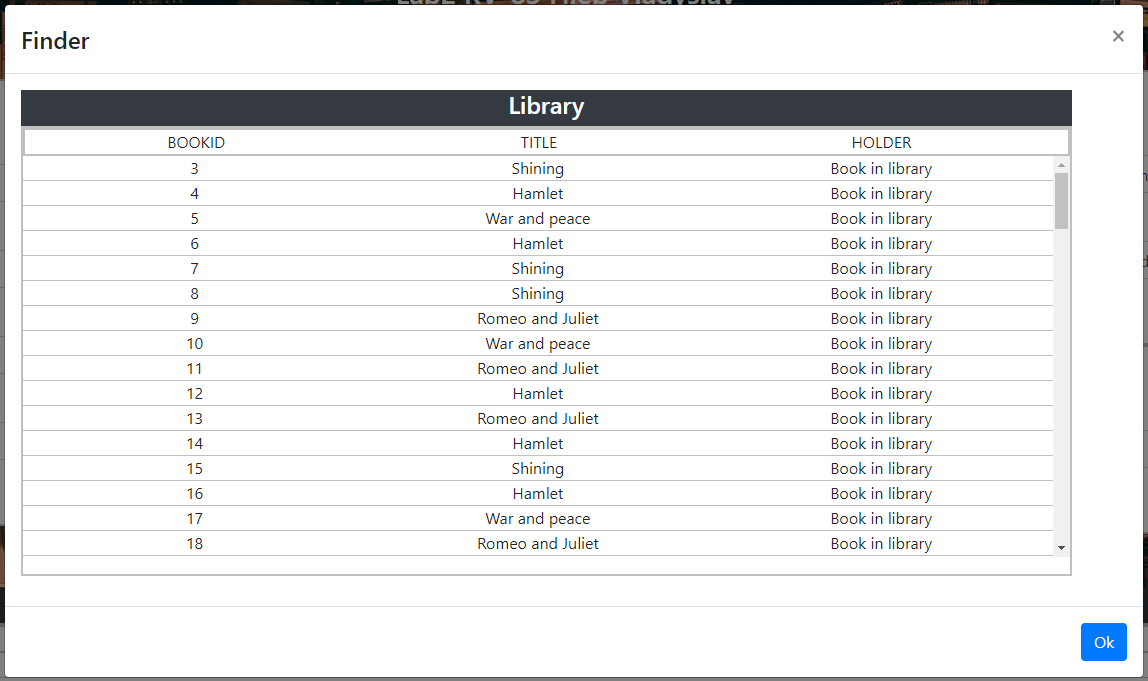
**Вибірка елементів з БД**

В формі “Finder” реалізована вибірка елементів з різних таблиць і за різними полями:

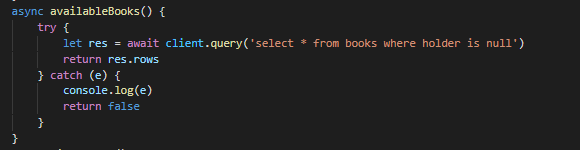


Р**ис 2.4 Форма Finder Рис 2.5 Випадаюче вікно форми Finder**

**1. “Get available books”** – вибирає книги, в яких holder = null. При находженні співпадінь програма повертає масив книг з їх даними:

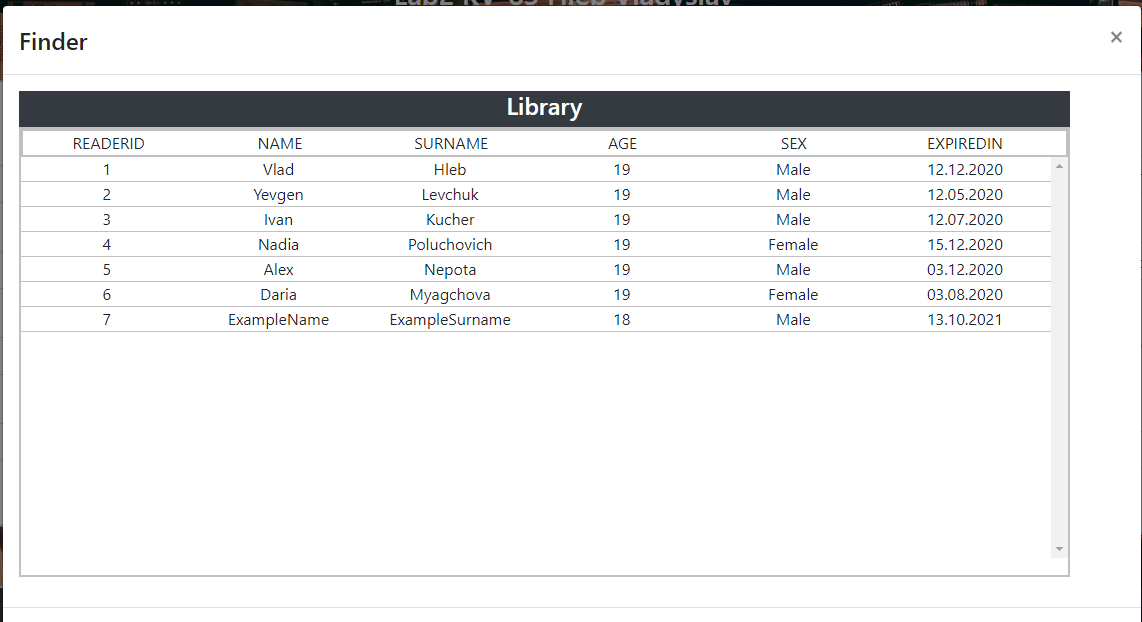


**Рис 2.6 Результат роботи “Get available books”**

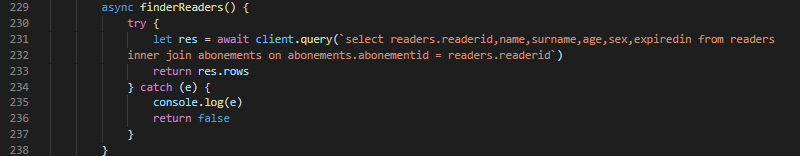
****

**Рис 2.7 Метод, який виводить всі записи в яких holder = null**

**2. “Get readers”** – виводить всіх читачів з бібліотеки:

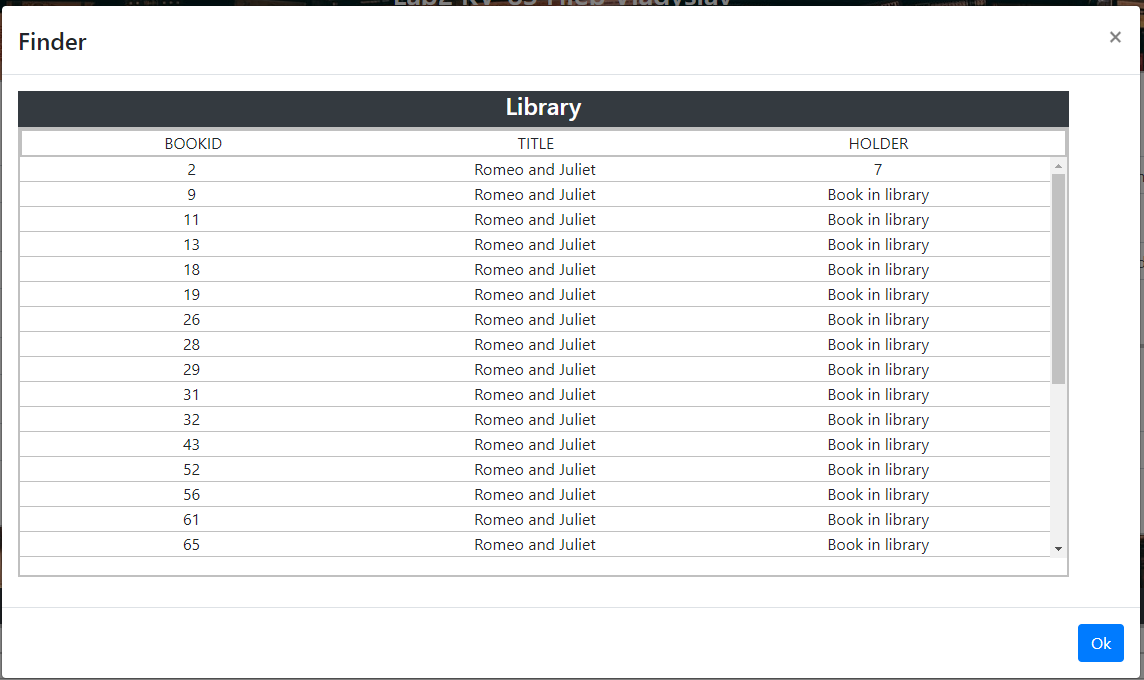


**Рис 2.8 Результат роботи “Get readers”**

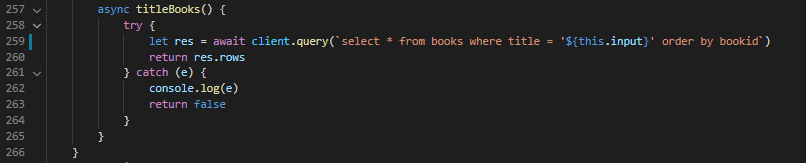


**Рис 2.9 Метод, який виводить записи з таблиці “Readers”**

**3. “Get books by title”** – вибирає книги з шуканою назвою:

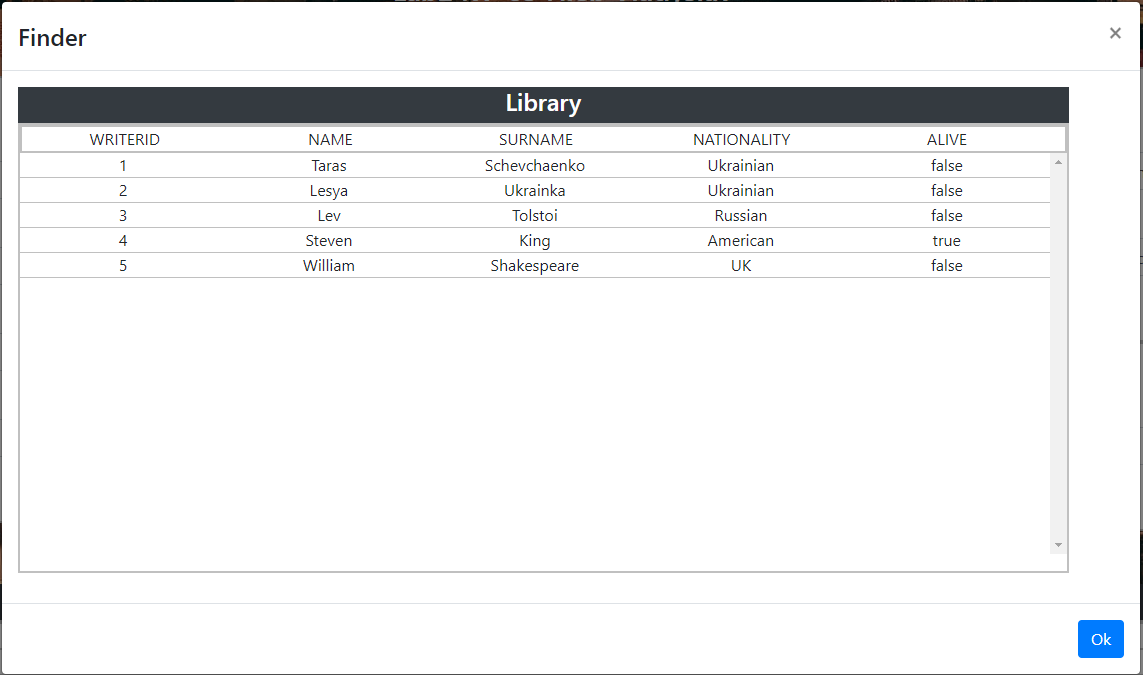
****

**Рис 2.9 Результат роботи методу “Get books by title”**

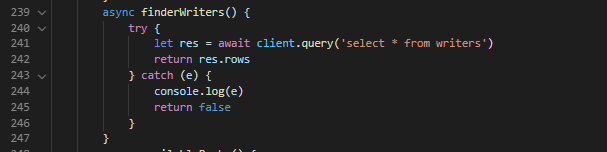
****

**Рис 2.10 Метод, який виводить всі записи з таблиці “Books” з введеною в формі назвою**

**4. “Get writers”** – виводить всіх письменників з бібліотеки:

****

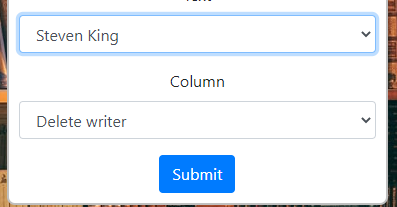
**Рис 2.11 Результат роботи “Get writers”**



**Рис 2.12 Метод, який виводить всі записи з таблиці “Writers”**

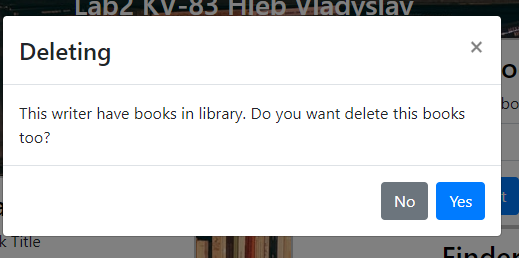
**Видалення зв’язаних між собою даних.**

В програмі реалізована можливість видалення автора.



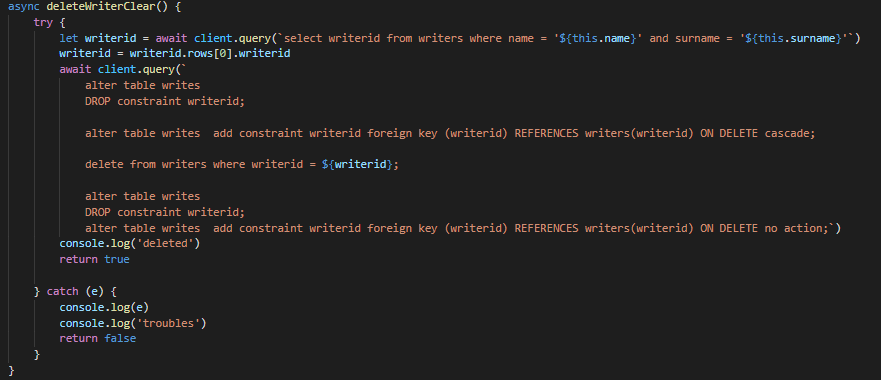
**Рис 2.13 Форма видалення Автора**

Оскільки таблиця Writers з’єднана з таблицею Writes і Books то видалення може бути здійснене тоді і тільки тоді, якщо в таблицях Writes і Books немає записів, які містять ключі цього автора (ON DELETE NO ACTION). В випадку коли була здійснена спроба видалити автора програма видає застереження.



**Рис 2.14 Підтвердження каскадного видалення**

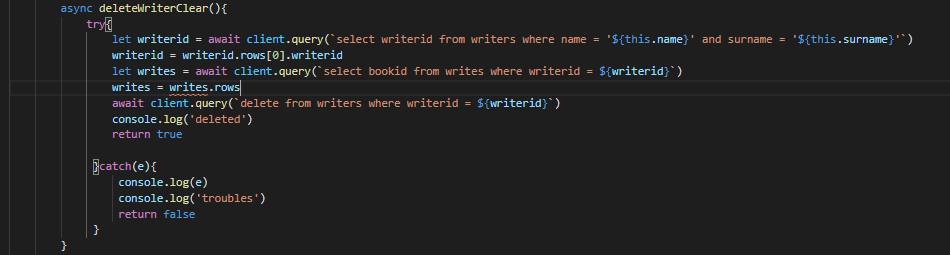
Якщо натиснути “Yes” програма змнить поле з NO ACTION на CASCADE і видалить всі дані про цього автора разом з книгами.

****

**Рис 2.15 Зміна режиму з NO ACTION на CASCADE**

PostgreSQL також дає можливість вибрати кілька режимів видалення:

1. **“ON DELETE SET NULL”** - всі Foreign key будуть мати значення Null, а за неможливості цього зробити буде повідомлення про помилку.
2. **“ON DELETE RESTRICT”** – не дає можливості видалити батьківський рядок, якщо в нього є дочірні. Поведінка дуже похожа на NO ACTION



**Рис 2.16 Метод очистки всіх даних режимом “CASCADE”**

**Навігація програми**

Програма створена за патерном MVC (Model-View-Controller). Складається з модулів Controller(index.js, router.js, controller.js), Model(tables.js). Роль View виконує користувацький веб-інтерфейс. Опис файлів:

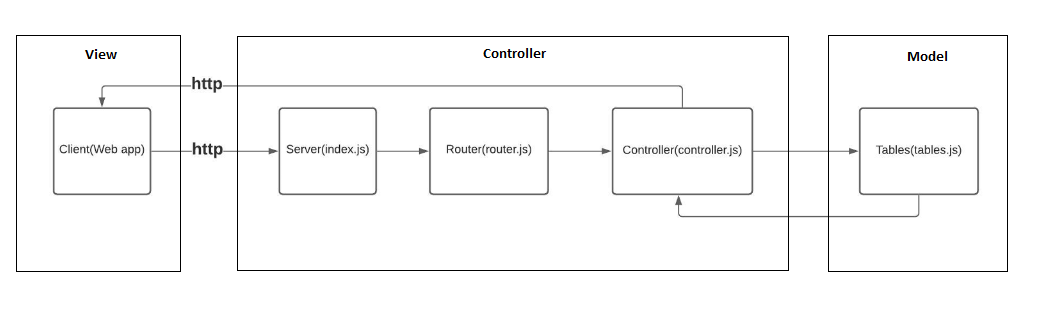
1. index.js (Рис 2.6) запускає сервер, підключає потрібні модулі і пакети
2. router.js (Рис 2.4) приймає запити з інтерфейсу користувача
3. controller.js(Рис 2.5) викликає потрібні методи для вибірки з БД
4. tables.js (Рис 2.6) посилає запити в БД і віддає відповідь контролеру

**Посилання для навігації по програмі**

1. [Index.js](#index)
   1. [(Line 25) Запуск сервера](#index)
2. [Router.js](#Router)
   1. ([Line 5) Роут, який приймає запит для отримання всіх існуючих книг з бібліотеки](#Router)
   2. [(Line 8) Роут, який приймає запит для отримання масиву авторів, книги яких є в бібліотеці](#Router)
   3. [(Line 11) Роут, який приймає запит для отримання масиву читачів, які зареєстровані в бібліотеці](#Router)
   4. [(Line 14) Роут, який приймає запит на отримання масиву книжок, які доступні для читання](#Router)
   5. [(Line 17) Роут, який приймає запит на пошук в бібліотеці за параметрами, описаними в тілі запиту](#Router)
   6. [(Line 20) Роут, який приймає запит на додавання користувача в бібліотеку](#Router)
   7. [(Line 23) Роут, який приймає запит на додавання певної кількості книг в бібліотеку. Кількість книг передається query-парамером.](#Router)
   8. [(Line 26) Роут, який приймає запит на видалення книги з бібліотеки. Ідентифікатор книги передаєтсья query-параметром](#Router)
   9. [(Line 29) Роут, який приймає запит на взяття книги з бібліотеки в користування читачем](#Router)
   10. [(Line 32) Роут, який приймає запит на додавання книги в бібліотеку](#Router)
3. [Controller.js](#Controller)
   1. [(Line 5) Метод контролера, який бере дані з таблиці Books і відправляє відповідь клієнту](#Controller)
   2. [(Line 9) Метод контролера, який бере дані з таблиці Readers і відправляє відповідь клієнту](#Controller)
   3. [(Line 13) Метод контролера, який бере дані з таблиці Writers і відправляє відповідь клієнту](#Controller)
   4. [(Line 18) Метод контролера, який бере дані з таблиці Books і відправляє відповідь клієнту](#Controller)
   5. [(Line 22) Метод контролера, який шукає співпадіння в таблицях і відправляє відповідь клієнту](#Controller)
   6. [(Line 27) Метод контролера, який додає запис в таблицю Readers](#Controller)
   7. [(Line 35) Метод контролера, який додає запис в таблицю Books](#Controller)
   8. [(Line 43) Метод контролера, який видаляє запис з таблиці Books](#Controller)
   9. [(Line 50) Метод контролера, який змінює запис в таблиці Books](#Controller_2)
   10. [(Line 56) Метод контролера, який додає запис в таблицю Books](#Controller_2)
4. [Tables.js](#Tables)
   1. [(Line 12) Метод класу Books, який створює запис в таблиці Books](#Tables_1)
   2. [(Line 17) Метод класу Books, який повертає ідентифікатор книги за назвою](#Tables_1)
   3. [(Line 26) Метод класу Books, який видаляє книгу за ідентифікатором](#Tables_1)
   4. [(Line 35) Метод класу Books, який повертає всі книги з бібліотеки](#Tables_2)
   5. [(Line 47) Метод класу Books, який повертає всі назви книжок з таблиці Books](#Tables_2)
   6. [(Line 56) Метод класу Books, який змінює запис в таблиці Books](#Tables_2)
   7. [(Line 65) Метод класу Books, який створює n записів в таблиці](#Tables_4)
   8. [(Line 89) Метод класу Writers, який повертає масив письменників](#Tables_4)
   9. [(Line 98) Метод класу Writers, який повертає ідентифікатор потрібного письменника](#Tables_4)
   10. [(Line 116) Метод класу Readers, який створює запис в таблиці Readers](#Tables_4)
   11. [(Line 131) Метод класу Readers, який повертає всі записи в таблиці Readers](#Tables_4)
   12. [(Line 140) Метод класу Readers, який повертає ідентифікатор читача](#Tables_4)
   13. [(Line 150) Метод класу Readers, який повертає всі записи в таблиці Readers в рядковому типі](#Tables_4)
   14. [(Line 168) Метод класу Finder, який провіряє активні поля в запиті на пошук](#Tables_5)
   15. [(Line 197) Метод класу Finder, який видаляє інформацію про письменника і всі його книги](#Tables_5)
   16. [(Line 219) Метод класу Finder, який видаляє інформацію про письменника](#Tables_5)
   17. [(Line 229) Метод класу Finder, який повертає всіх читачів](#Tables_5)
   18. [(Line 239) Метод класу Finder, який повертає всіх письменників](#Tables_5)

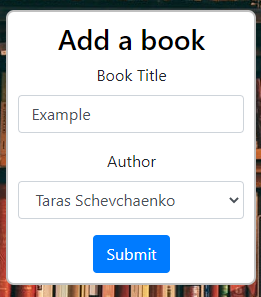
* 1. [(Line 248) Метод класу Finder, який повертає всі книжки, в яких немає користувача](#Tables_5)
  2. [(Line 257) Метод класу Finder, який шукає книги за назвою](#Tables_5)
  3. [(Line 272) Метод класу Writes, який створює запис в таблиці Writes](#Tables_5)

**Структура програми**

**Рис 2.17 Структура програми**

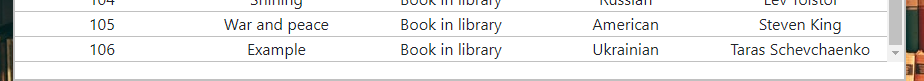
**Додавання запису в таблицю “Books”**

За додавання запису в таблицю Books відповідає форма “Add a book”.



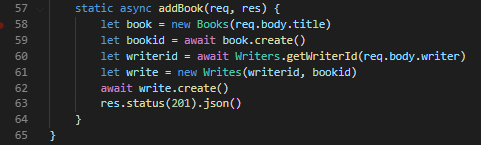
**Рис 2.18 Форма “Add a book”**

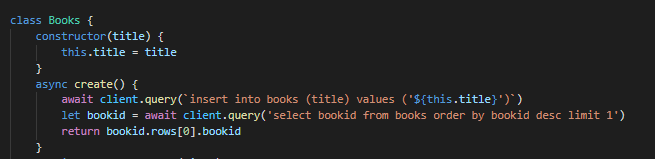
В формі можна ввести назву книжки і вибрати автора з вже доступних в бібліотеці. При підтвердженні форми книга добавиться в таблицю “Books”:



**Рис 2.19 Результат роботи форми “Add a book”**

**Метод, який додає книгу в бібліотеку**

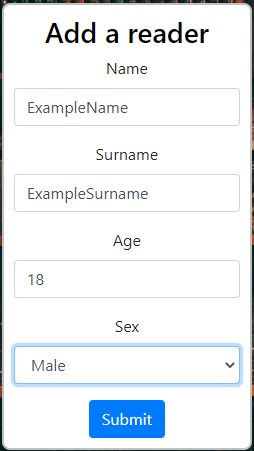




**Рис 2.20 Метод, який додає книгу в базу даних**

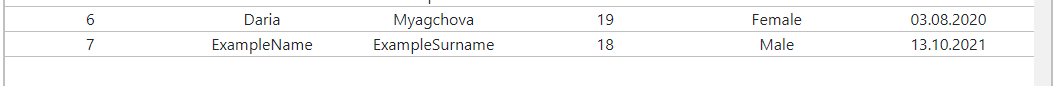
**Додавання запису в таблицю “Readers”**

За додавання запису в таблицю Readers відповідає форма “Add a reader”.

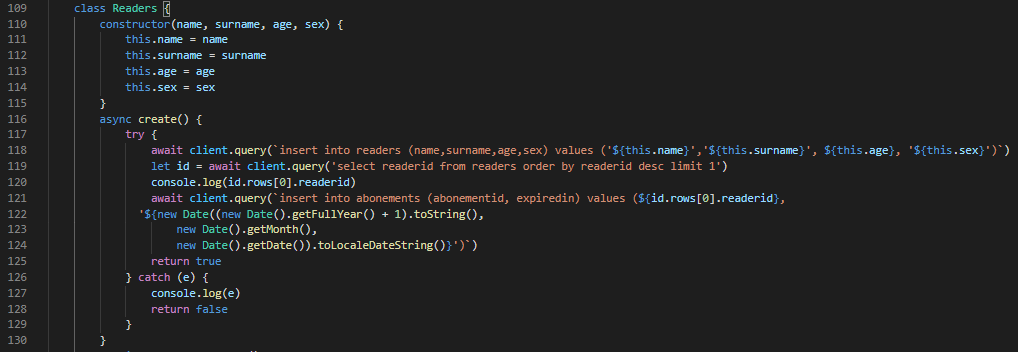


**Рис 2.21 Форма для додавання читача в таблицю “Readers”**

В формі можна ввести ім’я, фамілію, вік і стать.

При підтвердженні форми читач добавиться в таблицю “Readers”: 

**Рис 2.22 Результат роботи додавання читача в базу даних**

****

**Рис 2.23 Методи, які додають читача в базу даних і створюють абонемент(tables.js 116-129)**

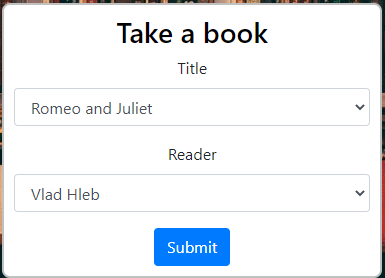
В разі помилки програма верне false (128 рядок).

**Зміна запису в таблиці “Readers”**

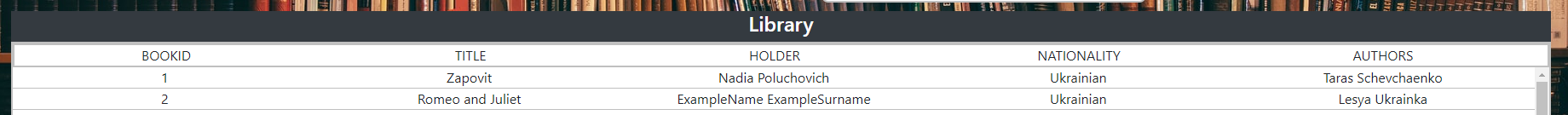
За зміну запису в таблиці Readers відповідає форма “Take a book”.

В формі можна ввести книгу і читача з випадаючого списку.

При підтвердженні форми в полі Holder появиться id читача , який взяв книгу:

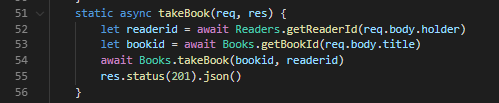


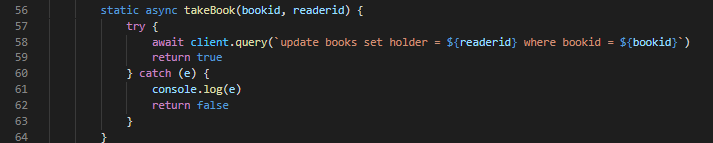
**Рис 2.24 Форма “Take a book”**



**Рис 2.24 Результат зміни запису, введеного в формі**

Метод, який змінює поле holder:

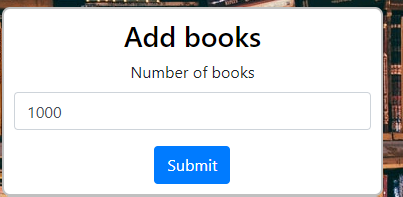




**Рис 2.25 Метод, який змінює поле holder**

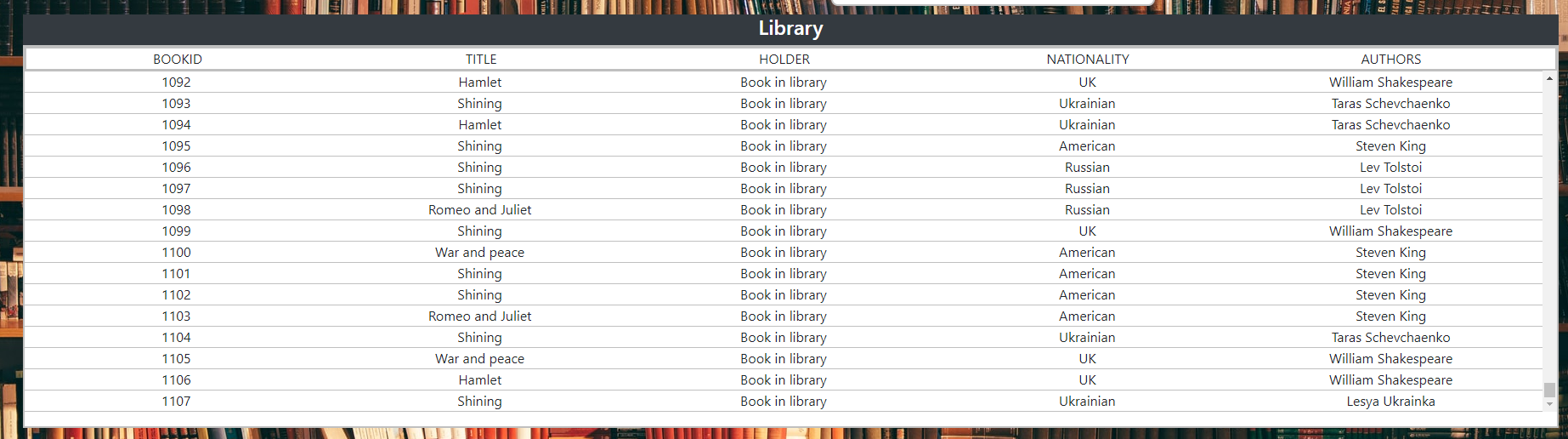
**Додавання N записів в таблицю “Books”**

За додавання записів відповідає Форма “Add books”

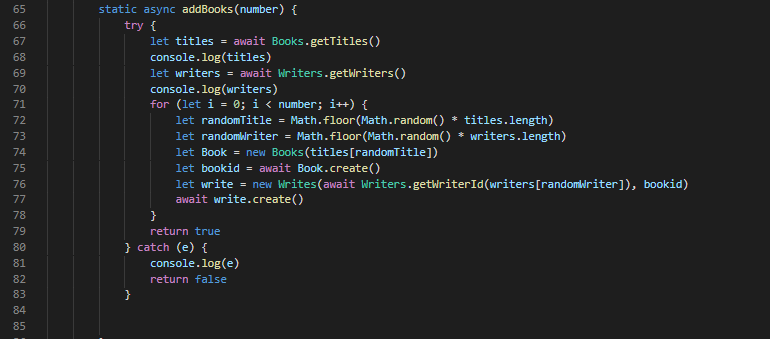


**Рис 2.26 Форма Add books**

В формі можна ввести потрібну кількість книг для додавання



**Рис 2.27 Згенеровані книги на основі вже існуючих книг**

****

**Рис 2.28 Метод, який додає книги**

**Лістинг модуля index.js**

****

**Рис 2.29 Модуль index.js**

**Лістинг модуля router.js**

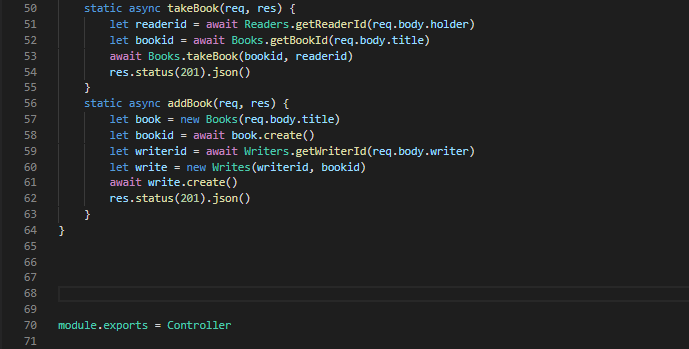
****

**Рис 2.30 Модуль router.js**

**Лістинг модуля controller.js**

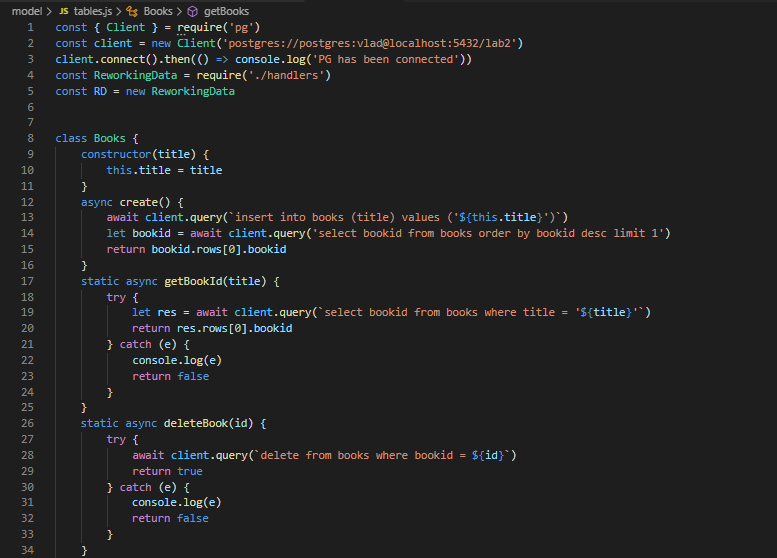
****

**Рис 2.31\_1 Модуль controller.js**

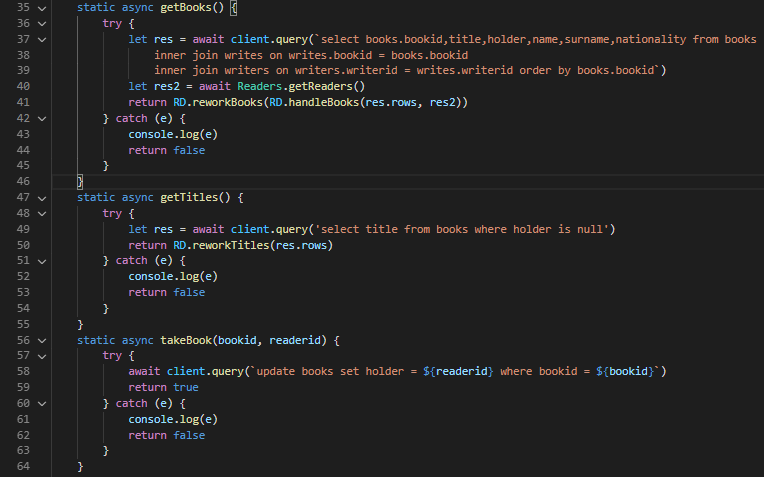
****

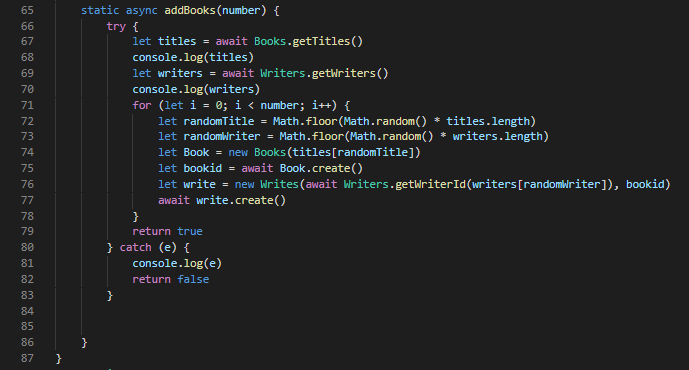
**Рис 2.31\_2 Модуль controller.js**

**Лістинг модуля tables.js**

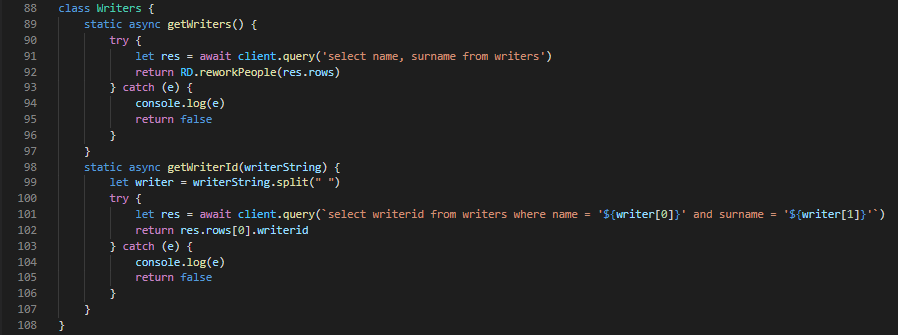


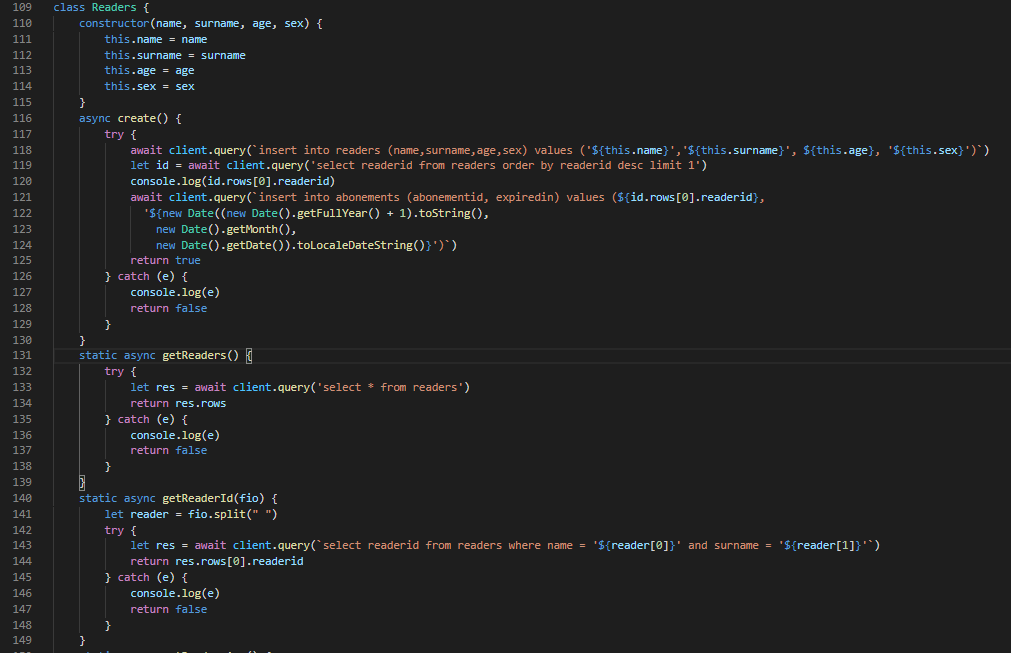
**Рис 2.32\_1 Модуль tables.js**

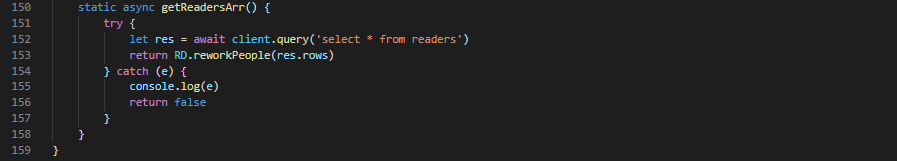




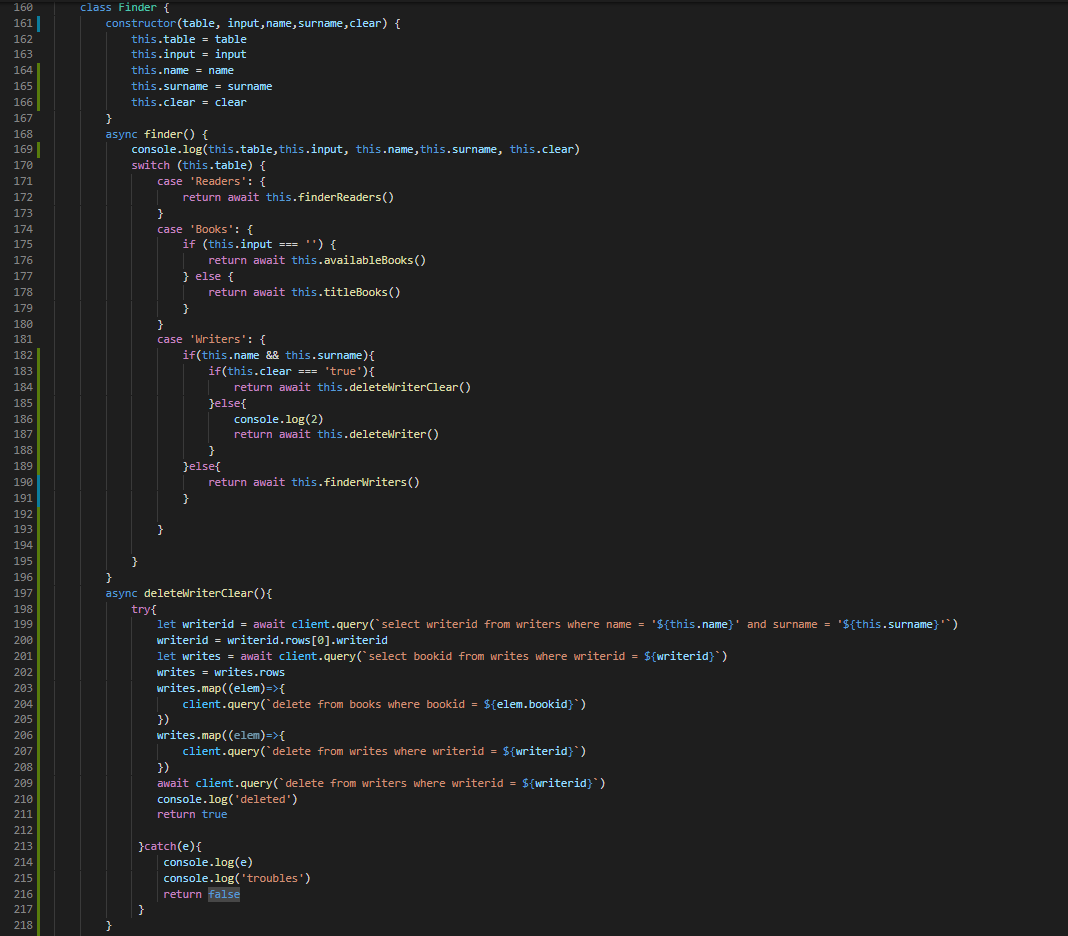
**Рис 2.32\_2 Модуль tables.js**







**Рис 2.32\_3 Модуль tables.js**

[](#Tables_4)

**Рис 2.32\_4 Модуль tables.js**



**Рис 2.32\_5 Модуль tables.js**