
Книжная библиотека

Выпуск 1.0.0

Бушуев Дмитрий

мар. 19, 2024

Оглавление

1 Основной модуль приложения (main)	1
2 Настройки приложения (settings)	3
3 Модули взаимодействия с базой данных	5
3.1 Модули моделей таблиц базы данных	5
3.1.1 Абстрактная модель таблицы - класс Table	5
3.1.2 Модель таблицы author - класс Author	6
3.1.3 Модель таблицы book - класс Book	8
3.1.4 Модель таблицы category - класс Category	12
3.1.5 Модель таблицы language - класс Language	13
3.1.6 Модель таблицы list_authors - класс ListAuthors	14
3.1.7 Модель таблицы list_notes - класс ListNotes	15
3.1.8 Модель таблицы list_states - класс ListStates	16
3.1.9 Модель таблицы list_tags - класс ListTags	19
3.1.10 Модель таблицы note - класс Note	20
3.1.11 Модель таблицы state - класс State	22
3.1.12 Модель таблицы tag - класс Tag	23
3.1.13 Модель таблицы user - класс User	24
3.1.14 Модель таблицы year - класс Year	26
3.2 Модули моделей представления базы данных	27
3.2.1 Абстрактная модель представления - класс View	27
3.2.2 Модель представления view_articles - класс ViewArticles	27
3.2.3 Модель представления view_books - класс ViewBooks	28
3.2.4 Модель представления view_end_read - класс ViewEndRead	29
3.2.5 Модель представления view_favorites - класс ViewFavorites	30
3.2.6 Модель представления view_filter - класс ViewFilter	31
3.2.7 Модель представления view_in_read - класс ViewInRead	34
3.2.8 Модель представления view_journals - класс ViewJournals	35
3.2.9 Модель представления view_search - класс ViewSearch	36
3.3 Класс CreateDataBase	37
3.4 Физическая модель базы данных	39
4 Книжная библиотека	41
4.1 Компоненты приложения	41
4.1.1 Компонент add_to_database	41
4.1.2 Компонент auth	44

4.1.3	Компонент book_read	45
4.1.4	Компонент change_book_information	47
4.1.5	Компонент change_state	50
4.1.6	Компонент function	51
4.1.7	Компонент search_filter	52
4.1.8	Компонент select_books	55
4.1.9	Компонент select_books_state	58
4.1.10	Компонент style	61
5	JavaScript код	63
5.1	add_file.js	63
5.2	change_file.js	65
5.3	change_state.js	70
5.4	filter_scripts.js	71
5.5	main_scripts.js	72
5.6	open_book.js	73
5.7	shift_style.js	75
6	CSS код	91
6.1	add_file.css	91
6.2	book_container.css	99
6.3	change_file.css	105
6.4	filter_window.css	108
6.5	main_book.css	109
6.6	open_book.css	114
6.7	user_menu.css	118
6.8	index_book.css	121
7	Стили приложения	125
	Содержание модулей Python	147

Основной модуль приложения (main)

Основной модуль приложения:

```
from settings import*
import sys
import database as db

if __name__ == '__main__':
    args = sys.argv[1:]
    if len(args) == 3:
        if args[2][-3:] == '.db':
            if args[0] == 'database' and args[1] == '-m':
                database=db.CreateDataBase(args[2])
                database.create_tables()
                database.create_views()
                database.insert_minimum_data_to_db()
            elif args[0] == 'database' and args[1] == '-t':
                database=db.CreateDataBase(args[2])
                database.create_tables()
                database.create_views()
                database.insert_data_to_db()
    else:
        app.run()
```

Команда создания базы данных с минимальным объемом данных:

```
python main.py database -m [название базы данных].db
```

Команда создания базы данных с тестовым объемом данных:

```
python main.py database -t [название базы данных].db
```

Версии используемых библиотек:

Название	Версия
Sphinx	7.2.6
sphinx-rtd-theme	2.0.0
sphinxcontrib-applehelp	1.0.8
sphinxcontrib-devhelp	1.0.6
sphinxcontrib-htmlhelp	2.0.5
sphinxcontrib-jsmath	1.0.1
sphinxcontrib-qthelp	1.0.7
sphinxcontrib-serializinghtml	1.1.10
blinker	1.7.0
certifi	2024.2.2
charset-normalizer	3.3.2
click	8.1.7
colorama	0.4.6
distlib	0.3.8
filelock	3.13.1
Flask	3.0.2
idna	3.6
importlib_metadata	7.0.2
itsdangerous	2.1.2
Jinja2	3.1.3
MarkupSafe	2.1.5
platformdirs	4.2.0
requests	2.31.0
setuptools	49.2.1
urllib3	2.2.1
virtualenv	20.25.1
Werkzeug	3.0.1
zipp	3.17.0

Настройки приложения (settings)

Основные параметры приложения:

Подключение blueprint к приложению:

```
app.register_blueprint(style) - отвечает за стили сайта
app.register_blueprint(change_state)
app.register_blueprint(select_books) - отвечает за выборку книг различных
    ↵категорий ("Журналы", "Книги", "Статьи")
app.register_blueprint(select_books_state)- отвечает за выборку книг различных
    ↵статусов ("На чтении", "Избранное", "Прочитанное")
app.register_blueprint(search_filter) - отвечает за выбор книг по фильтру
app.register_blueprint(book_read) - отвечает за страницу чтения книги
app.register_blueprint(add_to_database) - отвечает за добавление информации в
    ↵базу данных
app.register_blueprint(change_book_information) - отвечает за изменение
    ↵информации в базе данных
app.register_blueprint(auth) - отвечает за авторизацию на сайте
```

Переменные app.config:

```
app.config['DATABASE'] (str): Имя базы данных
app.config['ABOUT'] (About): Объект класса About (Таблица about)
app.config['AUTHOR'] (Author): Объект класса Author (Таблица author)
app.config['BOOK'] (Book): Объект класса Book (Таблица book)
app.config['CATEGORY'] (Category): Объект класса Category (Таблица category)
app.config['LANGUAGE'] (Language): Объект класса Language (Таблица language)
app.config['LIST_AUTHORS'] (ListAuthors): Объект класса ListAuthors (Таблица
    ↵list_authors)
app.config['LIST_NOTES'] (ListNotes): Объект класса ListNotes (Таблица list_
    ↵notes)
app.config['LIST_STATES'] (ListStates): Объект класса ListStates (Таблица list_
    ↵states)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
app.config['LIST_TAGS'] (ListTags): Объект класса ListTags (Таблица list_tags)
app.config['NOTE'] (Note): Объект класса Note (Таблица note)
app.config['STATE'] (State): Объект класса State (Таблица state)
app.config['TAG'] (Tag): Объект класса Tag (Таблица tag)
app.config['YEAR'] (Year): Объект класса Year (Таблица year)
app.config['USER'] (User): Объект класса User (Таблица user)
app.config['VIEW_ARTICLES'] (ViewArticles): Объект класса ViewArticles
    ↵(Представление view_articles)
app.config['VIEW_BOOKS'] (ViewBooks): Объект класса ViewBooks (Представление
    ↵view_books)
app.config['VIEW_END_READ'] (ViewEndRead): Объект класса ViewEndRead
    ↵(Представление view_end_read)
app.config['VIEW_IN_READ'] (ViewInRead): Объект класса ViewInRead
    ↵(Представление view_in_read)
app.config['VIEW_FAVORITES'] (ViewFavorites): Объект класса ViewFavorites
    ↵(Представление view_favorites)
app.config['VIEW_JOURNALS'] (ViewJornals): Объект класса ViewJornals
    ↵(Представление view_joornals)
app.config['VIEW_SEARCH'] (ViewSearch): Объект класса ViewSearch
    ↵(Представление search)
app.config['VIEW_FILTER'] (ViewFilter): Объект класса ViewFilter
    ↵(Представление filter)
app.config['CREATE_DATABASE'] (CreateDataBase): Объект класса CreateDataBase
app.config['SEARCH_TITLE'] (str): Переменная хранящая введенное название книги
    ↵для поиска (для перехода по страницам)
app.config['FILTER_YEAR'] (int): Переменная хранящая заданный год издания
    ↵фильтра (для перехода по страницам)
app.config['FILTER_TAG'] (str): Переменная хранящая заданный тэг фильтра (для
    ↵перехода по страницам)
app.config['FILTER_LANGUAGE'] (str): Переменная хранящая заданный язык фильтра
    ↵(для перехода по страницам)
app.config['USER_LOGIN'] (str): Логин пользователя
app.config['STYLE'] (str): Стиль сайта
```

Модули взаимодействия с базой данных

3.1 Модули моделей таблиц базы данных

3.1.1 Абстрактная модель таблицы - класс `Table`

```
class database.sqlite3_interface.tables.table.Table(name_db: str)
```

Базовые классы: ABC

Абстрактный класс `Table` - является моделью таблицы базы данных.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты: `:self.name_db (str)`: Имя базы данных

`Self.sqlite_connection (sqlite3.Connection)`

Соединение с базой данных

`Self.cursor (sqlite3.Cursor)`

Курсор базы данных

`Self.create_query (str)`

Запрос создания таблицы

`Self.query_last_row (str)`

Запрос получения id последней записи таблицы

`Self.data_to_db (list[list[str]])`

Тестовые данные для таблицы

`Self.query_insert (str)`

Запрос добавления данных в таблицу

`Self.query_select_all (str)`

Запрос выборки всех записей из таблицы

`close_connect_db() → None`

Функция закрывает соединение с базой данных.

`connect_db() → None`

Функция выполняет соединение с базой данных.

`create() → None`

Функция выполняет запрос создания представления в базе данных.

`get_data_to_insert() → list[list]`

Функция получения тестовых данных.

Результат

`list[list]: Тестовые данные`

`get_last_row() → int`

Функция получения id последней записи таблицы.

Результат

`int: id последней записи таблицы`

`insert(*args: list[list]) → None`

Функция Добавления тестовых данных в таблицу. :param *args: Список тестовых данных
:type *args: list[list]

`insert_data() → None`

Функция подготовки тестовых данных перед добавлением в таблицу.

`select_all() → list[list[str]]`

Функция выполняет выборку всех записей из представления базы данных.

Результат

`Результат выборки`

Тип результата

`list[list[str]]`

`update(query: str) → None`

Функция изменения информации в таблице.

Параметры

`query (str) – SQS-запрос изменения информации в базе данных`

3.1.2 Модель таблицы author - класс Author

`class database.sqlite3_interface.tables.author.Author(name_db: str)`

Базовые классы: `Table`

Класс является моделью таблицы author.

Параметры

`name_db (str) – Имя базы данных`

Атрибуты:

`self.create_query (str)`

Запрос создания таблицы author

```
self.query_insert (str)
    Запрос добавления данных в таблицу author
self.data_to_db (list[list[str]])
    Тестовые данные для таблицы author
self.query_select_all (str)
    Запрос выборки всех записей из таблицы author
self.query_select_on_id_book (str)
    Запрос выборки фио автора по id книги
self.query_select_on_full_name (str)
    Запрос выборки id автора по фио автора
```

SQL-запрос создания таблицы book (`self.create_query`):

```
CREATE TABLE IF NOT EXISTS author
(id_author INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
surname TEXTNULL,
name TEXT NULL,
patronymic TEXT NULL);
```

SQL-запрос (`self.query_insert`):

```
INSERT INTO author(name, surname, patronymic) VALUES (?, ?, ?);
```

SQL-запрос (`self.query_select_all`):

```
SELECT * FROM author;
```

`insert_get_data(surname: str, name: str, patronymic: str) → None`

Функция добавления данных в таблицу author

Параметры

- `surname (str)` – Фамилия автора
- `name (str)` – Имя автора
- `patronymic (str)` – Отчество автора

SQL-запрос (`self.query_insert`):

```
INSERT INTO author(name, surname, patronymic)
VALUES (?, ?, ?);
```

Результат

`list[list[str]]`: Результат выборки

`select_on_full_name(surname: str, name: str, patronymic: str) → list[list[str]]`

Функция выборки id автора по фио автора

Параметры

- `surname (str)` – Фамилия автора
- `name (str)` – Имя автора
- `patronymic (str)` – Отчество автора

SQL-запрос (`self.query_select_on_full_name`):

```
SELECT id_author FROM author
WHERE surname = ? AND name = ? AND patronymic = ?;
```

Результат

`list[list[str]]`: Результат выборки

`select_on_id_book(id_book: int) → list[list[str]]`

Функция выборки фио автора по id книги

Параметры

`id_book (int)` – id книги

SQL-запрос (`self.query_select_on_id_book`):

```
SELECT author.surname || ' ' || author.name
|| ' ' || author.patronymic AS fio FROM author
INNER JOIN list_authors
ON author.id_author = list_authors.id_author
INNER JOIN book
ON list_authors.id_book = book.id_book
WHERE book.id_book = ?;
```

Результат

`list[list[str]]`: Результат выборки

3.1.3 Модель таблицы book - класс Book

`class database.sqlite3_interface.tables.book.Book(name_db: str)`

Базовые классы: `Table`

Класс является моделью таблицы book.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания таблицы book

`self.query_insert (str)`

Запрос добавления данных в таблицу book

`self.data_to_db (list[list[str]])`

Тестовые данные для таблицы book

`self.query_select_all (str)`

Запрос выборки всех записей из таблицы book

`self.query_last_row (str)`

Запрос выборки последней записи таблицы book

`self.query_select_path_book (str)`

Запрос выборки пути к файлу книги по id книги

```

self.query_select_on_id_book (str)
    Запрос выборки данных о книге по id книги

self.query_update_title (str)
    Запрос изменения названия книги

self.query_update_page (str)
    Запрос изменения количества страниц книги

self.query_update_path_book (str)
    Запрос изменения пути к файлу книги

self.query_update_path_image (str)
    Запрос изменения пути к изображению книги

self.query_update_id_category (str)
    Запрос изменения категории книги

:self.query_update_id_language (str):Запрос изменения языка книги

self.query_update_id_year (str)
    Запрос изменения года издания книги

```

SQL-запрос создания таблицы book (self.create_query):

```

CREATE TABLE IF NOT EXISTS book
(id_book INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
title TEXT NOT NULL,
page INTEGER NULL,
path_book TEXT NULL,
path_image TEXT NULL,
id_year INTEGER,
id_language INTEGER,
id_about INTEGER,
id_category INTEGER,
FOREIGN KEY (id_year) REFERENCES year(id_year),
FOREIGN KEY (id_language) REFERENCES language(id_language),
FOREIGN KEY (id_about) REFERENCES about(id_about),
FOREIGN KEY (id_category) REFERENCES category(id_category));

```

SQL-запрос (self.query_insert):

```

INSERT INTO book (title,page, path_book, path_image,id_year,
id_language,id_about,id_category) VALUES(?,?,?,?,?,?,?,?,?);

```

SQL-запрос (self.query_select_all):

```

SELECT * FROM book;

```

SQL-запрос (self.query_last_row):

```

SELECT id_book FROM book ORDER BY id_book DESC;

```

`insert_get_data(title: str, page: int, path_book: str, path_image: str, id_year: int, id_language: int, id_about: int, id_category: int) → None`

Функция добавления данных в таблицу book

Параметры

- `title (str)` – Название книги
- `page (int)` – Количество страниц книги
- `path_book (str)` – Путь к файлу книги
- `path_image (str)` – Путь к изображению книги
- `id_year (int)` – id года книги
- `id_language (int)` – id языка книги
- `id_about (int)` – id описания книги
- `id_category (int)` – id категории книги

SQL-запрос (`self.query_insert`):

```
INSERT INTO book (title, page, path_book,
                  path_image, id_year, id_language,
                  id_about, id_category)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);
```

`select_on_id_book(id_book: int) → list[list[str]]`

Функция выборки данных о книге по id книги

Параметры

`id_book (int)` – id книги

SQL-запрос (`self.query_select_on_id_book`):

```
SELECT title, page, path_book, path_image, id_category,
       id_language, id_year FROM book WHERE id_book = ?;
```

Результат

`list[list[str]]`: Результат выборки

`select_path_book(id_book: int) → list[list[str]]`

Функция выборки пути к файлу книги по id книги

Параметры

`id_book (int)` – id книги

SQL-запрос (`self.query_select_path_book`):

```
SELECT path_book FROM book WHERE id_book = ?;
```

Результат

`list[list[str]]`: Результат выборки

`update_id_category(category: str, id_book: int) → None`

Функция изменения категории книги

Параметры

- `category (str)` – категория книги
- `id_book (int)` – id книги

SQL-запрос (self.query_update_id_category):

```
UPDATE book SET id_category = (SELECT id_category FROM category
                                 WHERE name = ?)
WHERE id_book = ?;
```

`update_id_language(language: str, id_book: int) → None`

Функция изменения языка книги

Параметры

- `language (str)` – язык книги
- `id_book (int)` – id книги

SQL-запрос (self.query_update_id_language):

```
UPDATE book SET id_language = (SELECT id_language FROM language
                                 WHERE name = ?)
WHERE id_book = ?;
```

`update_id_year(year: str, id_book: int) → None`

Функция изменения года издания книги

Параметры

- `year (str)` – год книги
- `id_book (int)` – id книги

SQL-запрос (self.query_update_id_year):

```
UPDATE book SET id_year = (SELECT id_year FROM year
                           WHERE year = ?)
WHERE id_book = ?;
```

`update_page(page: int, id_book: int) → None`

Функция изменения количества страниц книги

Параметры

- `page (int)` – Количество страниц книги
- `id_book (int)` – id книги

SQL-запрос (self.query_update_page):

```
UPDATE book SET page = ? WHERE id_book = ?;
```

`update_path_book(path_book: str, id_book: int) → None`

Функция изменения пути к файлу книги

Параметры

- `path_book (str)` – Путь к файлу книги
- `id_book (int)` – id книги

SQL-запрос (`self.query_update_path_book`):

```
UPDATE book SET path_book = ? WHERE id_book = ?;
```

`update_path_image(path_image: str, id_book: int) → None`

Функция изменения пути к изображению книги

Параметры

- `path_image (str)` – Путь к изображению книги
- `id_book (int)` – id книги

SQL-запрос (`self.query_update_path_image`):

```
UPDATE book SET path_image = ? WHERE id_book = ?;
```

`update_title(title: str, id_book: int) → None`

Функция изменения названия книги

Параметры

- `title (str)` – Название книги
- `id_book (int)` – id книги

SQL-запрос (`self.query_update_title`):

```
UPDATE book SET title = ? WHERE id_book = ?;
```

3.1.4 Модель таблицы category - класс Category

`class database.sqlite3_interface.tables.category.Category(name_db: str)`

Базовые классы: *Table*

Класс является моделью таблицы category.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания таблицы category

`self.query_insert (str)`

Запрос добавления данных в таблицу category

`self.data_to_db (list[list[str]])`

Тестовые данные для таблицы category

`self.query_select_all (str)`

Запрос выборки всех записей из таблицы category

`self.query_select_id_by_category (str)`

Запрос выборки id категории по названию категории из таблицы category

SQL-запрос создания таблицы category (`self.create_query`):

```
CREATE TABLE IF NOT EXISTS category
(id_category INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
name TEXT NULL);
```

SQL-запрос (self.query_insert):

```
INSERT INTO category(name) VALUES (?);
```

SQL-запрос (self.query_select_all):

```
SELECT * FROM category;
```

`select_id_by_category(category: str) → list[list[str]]`

Функция выборки id категории по названию категории из таблицы category

Параметры

`category (int)` – Название категории

SQL-запрос (self.query_select_id_by_category):

```
SELECT id_category FROM category
WHERE name = ?;
```

Результат

`list[list[str]]`: Результат выборки

3.1.5 Модель таблицы language - класс Language

`class database.sqlite3_interface.tables.language.Language(name_db: str)`

Базовые классы: *Table*

Класс является моделью таблицы language.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания таблицы language

`self.query_insert (str)`

Запрос добавления данных в таблицу language

`self.data_to_db (list[list[str]])`

Тестовые данные для таблицы language

`self.query_select_all (str)`

Запрос выборки всех записей из таблицы language

`self.query_select_id_by_language (str)`

Запрос выборки id языка по названию языку из таблицы language

SQL-запрос создания таблицы language (self.create_query):

```
CREATE TABLE IF NOT EXISTS language
(id_language INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
name TEXT NULL);
```

SQL-запрос (self.query_insert):

```
INSERT INTO language(name) VALUES (?);
```

SQL-запрос (self.query_select_all):

```
SELECT * FROM language;
```

select_id_by_language(language: str) → list[list[str]]

Функция выборки id языка по названию языку из таблицы language

Параметры

language (int) – Название языка

SQL-запрос (self.query_select_id_by_language):

```
SELECT id_language FROM language
WHERE name = ?;
```

Результат

list[list[str]]: Результат выборки

3.1.6 Модель таблицы list_authors - класс ListAuthors

```
class database.sqlite3_interface.tables.list_authors.ListAuthors(name_db: str)
```

Базовые классы: *Table*

Класс является моделью таблицы list_authors.

Параметры

name_db (str) – Имя базы данных

Атрибуты:

self.create_query (str)

Запрос создания таблицы list_authors

self.query_insert (str)

Запрос добавления данных в таблицу list_authors

self.data_to_db (list[list[str]])

Тестовые данные для таблицы list_authors

self.query_select_all (str)

Запрос выборки всех записей из таблицы list_authors

self.query_select_on_id_book_and_author (str)

Запрос выборки id списка по id автора и id книги из таблицы list_authors

SQL-запрос создания таблицы list_authors (self.create_query):

```
CREATE TABLE IF NOT EXISTS list_authors
(id_list_authors INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
id_author INTEGER NOT NULL,
id_book INTEGER NOT NULL,
FOREIGN KEY(id_book) REFERENCES books(id_book),
FOREIGN KEY(id_author) REFERENCES author(id_author));
```

SQL-запрос (self.query_select_all):

```
SELECT * FROM list_authors;
```

`insert_get_date(id_author: int, id_book: int) → None`

Функция добавления данных в таблицу list_authors

Параметры

- `id_author (int)` – id автора
- `id_book (int)` – id книги

SQL-запрос (self.query_insert):

```
INSERT INTO list_authors(id_author, id_book)
VALUES (?, ?);
```

`select_on_id_book_and_author(id_author: int, id_book: int) → list[list[str]]`

Функция выборки id списка по id автора и id книги из таблицы list_authors

Параметры

- `id_author (int)` – id автора
- `id_book (int)` – id книги

SQL-запрос (self.query_select_on_id_book_and_author):

```
SELECT id_list_authors FROM list_authors
WHERE id_author = ? AND id_book = ?
```

Результат

`list[list[str]]`: Результат выборки

3.1.7 Модель таблицы list_notes - класс ListNotes

```
class database.sqlite3_interface.tables.list_notes.ListNotes(name_db: str)
```

Базовые классы: `Table`

Класс является моделью таблицы list_notes.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

```
self.create_query (str)
    Запрос создания таблицы list_notes

self.query_insert (str)
    Запрос добавления данных в таблицу list_notes

self.data_to_db (list[list[str]])
    Тестовые данные для таблицы list_notes

self.query_select_all (str)
    Запрос выборки всех записей из таблицы list_notes
```

SQL-запрос создания таблицы list_notes (self.create_query):

```
CREATE TABLE IF NOT EXISTS list_notes
(id_list_notes INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
id_note INTEGER NOT NULL,
id_book INTEGER NOT NULL,
id_user INTEGER NOT NULL,
FOREIGN KEY(id_book) REFERENCES book(id_book),
FOREIGN KEY(id_user) REFERENCES user(id_user),
FOREIGN KEY(id_note) REFERENCES note(id_note));
```

SQL-запрос (self.query_select_all):

```
SELECT * FROM list_notes;
```

insert_get_data(id_note: int, id_book: int, id_user: int) → None

Функция добавления данных в таблицу list_notes

Параметры

- id_note (*int*) – id заметки
- id_book (*int*) – id книги
- id_user (*int*) – id пользователя

SQL-запрос (self.query_insert):

```
INSERT INTO list_notes(id_note, id_book, id_user)
VALUES (?, ?, ?);
```

3.1.8 Модель таблицы list_states - класс ListStates

class database.sqlite3_interface.tables.list_states.ListStates(name_db: str)

Базовые классы: *Table*

Класс является моделью таблицы list_states.

Параметры

name_db (*str*) – Имя базы данных

Атрибуты:

```
self.create_query (str)
    Запрос создания таблицы list_states
```

```

self.query_insert (str)
    Запрос добавления данных в таблицу list_states

self.data_to_db (list[list[str]])
    Тестовые данные для таблицы list_states

self.query_select_all (str)
    Запрос выборки всех записей из таблицы list_states

self.query_select_state_page (str)
    Запрос выборки страницы для закладки по id книги и id пользователя

self.query_update_state_page (str)
    Запрос изменения страницы закладки книги

self.query_update_state (str)
    Запрос изменения статуса книги

self.query_select_user_books (str)
    Запрос выборки книг пользователя

```

SQL-запрос создания таблицы list_states (self.create_query):

```

CREATE TABLE IF NOT EXISTS list_states
(id_list_states INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
page INTEGER NULL,
id_state INTEGER NOT NULL,
id_book INTEGER NOT NULL,
id_user INTEGER NOT NULL,
FOREIGN KEY(id_book) REFERENCES book(id_book),
FOREIGN KEY(id_user) REFERENCES user(id_user),
FOREIGN KEY(id_state) REFERENCES state(id_state));

```

SQL-запрос (self.query_select_all):

```
SELECT * FROM list_states;
```

insert_get_data(page: int, id_state: int, id_book: int, id_user: int) → None

Функция добавления данных в таблицу list_states

Параметры

- **page (int)** – Номер страницы
- **id_state (int)** – id статуса книги
- **id_book (int)** – id книги
- **id_user (int)** – id пользователя

SQL-запрос (self.query_update_state):

```

INSERT INTO list_states(page, id_state, id_book, id_user)
VALUES (?, ?, ?, ?);

```

select_state_page(id_book: int, id_user: int) → list[list[str]]

Функция выборки страницы для закладки по id книги и id пользователя

Параметры

- `id_book (int)` – id книги
- `id_user (int)` – id пользователя

SQL-запрос (`self.query_select_state_page`):

```
SELECT page FROM list_states  
WHERE id_book = ? AND id_user = ?;
```

Результат

`list[list[str]]`: Результат выборки

`select_user_books(login: str) → list[list[str]]`

Функция выборки книг пользователя

Параметры

`login (int)` – Логин пользователя

SQL-запрос (`self.query_select_user_books`):

```
SELECT id_book FROM list_states  
INNER JOIN user ON list_states.id_user = user.id_user  
INNER JOIN state ON state.id_state = list_states.id_state  
WHERE user.login = ?;
```

Результат

`list[list[str]]`: Результат выборки

`update_state(id_state: int, id_book: int, id_user: int) → None`

Функция изменения статуса книги

Параметры

- `id_state (int)` – id статуса книги
- `id_book (int)` – id книги
- `id_user (int)` – id пользователя

SQL-запрос (`self.query_update_state`):

```
UPDATE list_states SET id_state = ?  
WHERE id_book = ? AND id_user = ?;
```

`update_state_page(new_page: int, id_book: int, id_user: int) → None`

Функция изменения страницы закладки книги

Параметры

- `new_page (int)` – Новая страница
- `id_book (int)` – id книги
- `id_user (int)` – id пользователя

SQL-запрос (`self.query_update_state_page`):

```
UPDATE list_states SET page = ?
WHERE id_book = ? AND id_user = ?;
```

3.1.9 Модель таблицы list_tags - класс ListTags

class database.sqlite3_interface.tables.list_tags.ListTags(name_db: str)

Базовые классы: *Table*

Класс является моделью таблицы list_tags.

Параметры

name_db (str) – Имя базы данных

Атрибуты:

self.create_query (str)

Запрос создания таблицы list_tags

self.query_insert (str)

Запрос добавления данных в таблицу list_tags

self.data_to_db (list[list[str]])

Тестовые данные для таблицы list_tags

self.query_select_all (str)

Запрос выборки всех записей из таблицы list_tags

self.query_select_on_id_book_and_tag (str)

Запрос выборки id по id книги и id тэгу

SQL-запрос создания таблицы list_tags (*self.create_query*):

```
CREATE TABLE IF NOT EXISTS list_tags
(id_list_tags INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
id_tag INTEGER NOT NULL,
id_book INTEGER NOT NULL,
FOREIGN KEY(id_book) REFERENCES book(id_book),
FOREIGN KEY(id_tag) REFERENCES tag(id_tag));
```

SQL-запрос (*self.query_select_all*):

```
SELECT * FROM list_tags;
```

insert_get_date(*id_tag*: int, *id_book*: int) → None

Функция добавления данных в таблицу

Параметры

- *id_tag* (int) – id тэга

- *id_book* (int) – id книги

SQL-запрос (*self.query_insert*):

```
INSERT INTO list_tags(id_tag, id_book) VALUES (?, ?);
```

`select_on_id_book_and_tag(id_tag: int, id_book: int) → list[list[str]]`

Функция выборки id по id книги и id тэгу

Параметры

- `id_tag (int)` – id тэга
- `id_book (int)` – id книги

SQL-запрос (`self.query_select_on_id_book_and_tag`):

```
SELECT id_list_tags FROM list_tags
WHERE id_tag = ? AND id_book = ?;
```

Результат

`list[list[str]]`: Результат выборки

3.1.10 Модель таблицы note - класс Note

`class database.sqlite3_interface.tables.note.Note(name_db: str)`

Базовые классы: `Table`

Класс является моделью таблицы note.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания таблицы note

`self.query_insert (str)`

Запрос добавления данных в таблицу note

`self.query_last_row (str)`

Получение id последней записи таблицы note

`self.data_to_db (list[list[str]])`

Тестовые данные для таблицы note

`self.query_select_all (str)`

Запрос выборки всех записей из таблицы note

`self.query_select_on_id_book (str)`

Запрос выборки заметок для книги по id книги

`self.query_select_on_id_user_book (str)`

Запрос выборки заметок для книги для заданного пользователя

SQL-запрос создания таблицы note (`self.create_query`):

```
CREATE TABLE IF NOT EXISTS note
(id_note INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
text_note TEXT NULL);
```

SQL-запрос (`self.query_select_all`):

```
SELECT * FROM note;
```

SQL-запрос (self.query_last_row):

```
SELECT id_note FROM note ORDER BY id_note DESC;
```

`insert_get_data(text_note: str) → None`

Функция добавления заметки для книги

Параметры

- text_note (*int*) – Текст заметки

SQL-запрос (self.query_insert):

```
INSERT INTO note(text_note) VALUES (?);
```

`select_on_id_book(id_book: int) → list[list[str]]`

Функция выборки заметок для книги по id книги

Параметры

- id_book (*int*) – id книги

SQL-запрос (self.query_select_on_id_book):

```
SELECT note.id_note, note.text_note FROM note
INNER JOIN list_notes ON list_notes.id_note = note.id_note
INNER JOIN book ON list_notes.id_book = book.id_book
WHERE book.id_book = ?;
```

Результат

list[list[str]]: Результат выборки

`select_on_id_user_book(id_book: int, id_user: int) → list[list[str]]`

Функция выборки заметок для книги для заданного пользователя

Параметры

- id_book (*int*) – id книги
- id_user (*int*) – id пользователя

SQL-запрос (self.query_select_on_id_user_book):

```
SELECT * FROM note WHERE id_note
IN (SELECT id_note FROM list_notes
WHERE id_book = ? AND id_user = ?);
```

Результат

list[list[str]]: Результат выборки

3.1.11 Модель таблицы state - класс State

```
class database.sqlite3_interface.tables.state.State(name_db: str)
```

Базовые классы: *Table*

Класс является моделью таблицы state.

Параметры

name_db (str) – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания таблицы state

`self.query_insert (str)`

Запрос добавления данных в таблицу state

`self.data_to_db (list[list[str]])`

Тестовые данные для таблицы state

`self.query_select_all (str)`

Запрос выборки всех записей из таблицы state

`self.query_select_state_name (str)`

Запрос выборки статуса книги для пользователя

SQL-запрос создания таблицы state (`self.create_query`):

```
CREATE TABLE IF NOT EXISTS state
(id_state INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
state_name TEXT NULL);
```

SQL-запрос (`self.query_insert`):

```
INSERT INTO state(state_name) VALUES (?);
```

SQL-запрос (`self.query_select_all`):

```
SELECT * FROM state;
```

`select_state_name(id_user: int, id_book: int) → str`

Функция выборки статуса книги для пользователя

Параметры

- `id_user (int)` – id пользователя
- `id_book (int)` – id книги

SQL-запрос (`self.query_select_state_name`):

```
SELECT state_name FROM state WHERE id_state
IN (SELECT id_state FROM list_states
WHERE id_user = ? AND id_book = ?);
```

Результат

`state_name: Статус книги для пользователя`

3.1.12 Модель таблицы tag - класс Tag

```
class database.sqlite3_interface.tables.tag.Tag(name_db: str)
```

Базовые классы: *Table*

Класс является моделью таблицы tag.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания таблицы tag

`self.query_insert (str)`

Запрос добавления данных в таблицу tag

`self.data_to_db (list[list[str]])`

Тестовые данные для таблицы tag

`self.query_select_all (str)`

Запрос выборки всех записей из таблицы tag

`self.query_select_on_id_book (str)`

Запрос выборки списка тэгов для книги по id книги

`self.query_select_id_on_tag (str)`

Запрос выборки id тэга по названию тэга

SQL-запрос создания таблицы tag (`self.create_query`):

```
CREATE TABLE IF NOT EXISTS tag
(id_tag INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
tag_name TEXT NULL);
```

SQL-запрос (`self.query_insert`):

```
INSERT INTO tag(tag_name) VALUES (?);
```

SQL-запрос (`self.query_select_all`):

```
SELECT * FROM tag;
```

`insert_get_data(tag_name: str) → None`

Функция добавления тэга в базу данных

Параметры

`tag_name (int)` – название тэга

SQL-запрос (`self.query_insert`):

```
INSERT INTO tag(tag_name) VALUES (?);
```

`select_id_on_tag(tag_name: str) → list[list[str]]`

Функция выборки id тэга по названию тэга

Параметры

`tag_name (int)` – название тэга

SQL-запрос (self.query_select_id_on_tag):

```
SELECT id_tag FROM tag WHERE tag_name = ?;
```

Результат

Результат выборки

Тип результата

`list[list[str]]`

`select_on_id_book(id_book: int) → list[list[str]]`

Функция выборки названия тэга по id книги

Параметры

`id_book (int)` – id книги

SQL-запрос (self.query_select_on_id_book):

```
SELECT tag_name FROM tag
INNER JOIN list_tags ON tag.id_tag = list_tags.id_tag
INNER JOIN book ON list_tags.id_book = book.id_book
WHERE book.id_book = ?;
```

Результат

Результат выборки

Тип результата

`list[list[str]]`

3.1.13 Модель таблицы user - класс User

```
class database.sqlite3_interface.tables.user.User(name_db: str)
```

Базовые классы: *Table*

Класс является моделью таблицы user.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания таблицы user

`self.query_insert (str)`

Запрос добавления данных в таблицу user

`self.data_to_db (list[list[str]])`

Тестовые данные для таблицы user

`self.query_select_all (str)`

Запрос выборки всех записей из таблицы user

```
self.query_select_id_on_login(str)
    Запрос выборки id пользователя с заданным логином

self.query_select_password_on_login(str)
    Запрос выборки пароля пользователя с заданным логином
```

SQL-запрос создания таблицы user (self.create_query):

```
CREATE TABLE IF NOT EXISTS user
(id_user INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
login TEXT NOT NULL,
password TEXT NOT NULL);
```

SQL-запрос (self.query_insert):

```
INSERT INTO user(login, password) VALUES (?, ?);
```

SQL-запрос (self.query_select_all):

```
SELECT * FROM user;
```

select_id_on_login(login: str) → list[list[str]]

Функция получения id пользователя по логину пользователя

Параметры

login (*int*) – Логин пользователя

SQL-запрос (self.query_select_id_on_login):

```
SELECT id_user FROM user WHERE login = ?;
```

Результат

Результат выборки

Тип результата

list[list[str]]

select_password_on_login(login: str) → list[list[str]]

Функция получения пароля пользователя по логину пользователя

Параметры

login (*int*) – Логин пользователя

SQL-запрос (self.query_select_password_on_login):

```
SELECT password FROM user WHERE login = ?;
```

Результат

Результат выборки

Тип результата

list[list[str]]

3.1.14 Модель таблицы year - класс Year

```
class database.sqlite3_interface.tables.year.Year(name_db: str)
```

Базовые классы: *Table*

Класс является моделью таблицы year.

Параметры

name_db (str) – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания таблицы year

`self.query_insert (str)`

Запрос добавления данных в таблицу year

`self.data_to_db (list[list[str]])`

Тестовые данные для таблицы year

`self.query_select_all (str)`

Запрос выборки всех записей из таблицы year

`self.query_select_id_by_year (str)`

Запрос выборки id года с заданным годом

SQL-запрос создания таблицы year (`self.create_query`):

```
CREATE TABLE IF NOT EXISTS year
(id_year INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
year TEXT NULL);
```

SQL-запрос (`self.query_insert`):

```
INSERT INTO year(year) VALUES (?);
```

SQL-запрос (`self.query_select_all`):

```
SELECT * FROM year;
```

`select_id_by_year(year: int) → list[list[str]]`

Функция получения id года по заданному году

Параметры

year (int) – Год издания

SQL-запрос (`self.query_select_id_by_year`):

```
SELECT id_year FROM year WHERE year = ?;
```

Результат

Результат выборки

Тип результата

`list[list[str]]`

3.2 Модули моделей представления базы данных

3.2.1 Абстрактная модель представления - класс View

```
class database.sqlite3_interface.views.view.View(name_db: str)
```

Базовые классы: ABC

Абстрактный класс View - является моделью представления базы данных.

Параметры

name_db (str) – Имя базы данных

Атрибуты:

self.name_db (str)

Имя базы данных

self.sqlite_connection (sqlite3.Connection)

Соединение с базой данных

self.cursor (sqlite3.Cursor)

Курсор базы данных

self.create_query (str)

Запрос создание представления

self.query_select_all (str)

Запрос выборки всех записей из представления

close_connect_db() → None

Функция закрывает соединение с базой данных.

connect_db() → None

Функция выполняет соединение с базой данных.

create() → None

Функция выполняет запрос создания представления в базе данных.

select_all() → list[list[str]]

Функция выполняет выборку всех записей из представления базы данных.

Результат

Результат выборки

Тип результата

list[list[str]]

3.2.2 Модель представления view_articles - класс ViewArticles

```
class database.sqlite3_interface.views.view_articles.ViewArticles(name_db: str)
```

Базовые классы: *View*

Класс является моделью представления articles_category.

Параметры

name_db (str) – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания представления

`self.query_select_all (str)`

Запрос выборки всех записей представления

`self.query_select_for_user (str)`

Запрос выборки всех записей представления для пользователя

SQL-запрос создания представления filter (`self.create_query`):

```
CREATE VIEW IF NOT EXISTS articles_category AS
SELECT book.id_book, book.title, book.path_book,
book.path_image, about.information, language.name AS language,
year.year, category.name FROM book
INNER JOIN about ON about.id_book = book.id_about
INNER JOIN language ON language.id_language = book.id_language
INNER JOIN category ON category.id_category = book.id_category
INNER JOIN year ON year.id_year = book.id_year
WHERE category.name='Статьи'
```

SQL-запрос (`self.query_select_all`):

```
SELECT * FROM articles_category;
```

SQL-запрос (`self.query_select_for_user`):

```
SELECT * FROM articles_category WHERE login = ?;
```

3.2.3 Модель представления `view_books` - класс `ViewBooks`

`class database.sqlite3_interface.views.view_books.ViewBooks(name_db: str)`

Базовые классы: `View`

Класс является моделью представления `books_category`.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания представления

`self.query_select_all (str)`

Запрос выборки всех записей представления

`self.query_select_for_user (str)`

Запрос выборки всех записей представления для пользователя

SQL-запрос создания представления filter (`self.create_query`):

```
CREATE VIEW IF NOT EXISTS books_category AS
SELECT book.id_book, book.title, book.path_book,
book.path_image, about.information, language.name AS language,
year.year, category.name FROM book
```

(continues on next page)

(продолжение с предыдущей страницы)

```
INNER JOIN about ON about.id_book = book.id_about
INNER JOIN language ON language.id_language = book.id_language
INNER JOIN category ON category.id_category = book.id_category
INNER JOIN year ON year.id_year = book.id_year
WHERE category.name='Книги' ;
```

SQL-запрос (self.query_select_all):

```
SELECT * FROM books_category;
```

SQL-запрос (self.query_select_for_user):

```
SELECT * FROM books_category WHERE login = ?;
```

3.2.4 Модель представления view_end_read - класс ViewEndRead

```
class database.sqlite3_interface.views.view_end_read.ViewEndRead(name_db: str)
```

Базовые классы: *View*

Класс является моделью представления state_end_read.

Параметры

name_db (str) – Имя базы данных

Атрибуты:**self.create_query (str)**

Запрос создания представления

self.query_select_all (str)

Запрос выборки всех записей представления

self.query_select_for_user (str)

Запрос выборки всех записей представления для пользователя

SQL-запрос создания представления filter (self.create_query):

```
CREATE VIEW IF NOT EXISTS state_end_read AS
SELECT book.id_book, book.title, book.path_book,
book.path_image, about.information, language.name AS language,
year.year, category.name, user.id_user,
user.login, state.state_name FROM book
INNER JOIN about ON about.id_book = book.id_about
INNER JOIN language ON language.id_language = book.id_language
INNER JOIN category ON category.id_category = book.id_category
INNER JOIN year ON year.id_year = book.id_year
INNER JOIN list_states ON list_states.id_book = book.id_book
INNER JOIN state ON list_states.id_state = state.id_state
INNER JOIN user ON list_states.id_user = user.id_user
WHERE state.state_name = 'Прочитанное' ;
```

SQL-запрос (self.query_select_all):

```
SELECT * FROM state_end_read;
```

`select_for_user(login: str) → None`

Функция выполняет выборку записей со статусом «Прочитанное» для пользователя

Параметры

`login (str)` – Логин пользователя

SQL-запрос (`self.query_select_for_filter_all`):

```
SELECT * FROM state_end_read WHERE login = ?;
```

Результат

Результаты выборки по фильтру

Тип результата

`list[list[str]]`

3.2.5 Модель представления `view_favorites` – класс `ViewFavorites`

```
class database.sqlite3_interface.views.view_favorites.ViewFavorites(name_db: str)
```

Базовые классы: `View`

Класс является моделью представления `state_favorites`.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания представления

`self.query_select_all (str)`

Запрос выборки всех записей представления

`self.query_select_for_user (str)`

Запрос выборки всех записей представления для пользователя

SQL-запрос создания представления `filter (self.create_query)`:

```
CREATE VIEW IF NOT EXISTS stateFavorites AS
SELECT book.id_book, book.title, book.path_book,
book.path_image, about.information, language.name AS language,
year.year, category.name, user.id_user,
user.login, state.state_name FROM book
INNER JOIN about ON about.id_book = book.id_about
INNER JOIN language ON language.id_language = book.id_language
INNER JOIN category ON category.id_category = book.id_category
INNER JOIN year ON year.id_year = book.id_year
INNER JOIN list_states ON list_states.id_book = book.id_book
INNER JOIN state ON list_states.id_state = state.id_state
INNER JOIN user ON list_states.id_user = user.id_user
WHERE state.state_name = 'Избранное'
```

SQL-запрос (self.query_select_all):

```
SELECT * FROM stateFavorites;
```

select_for_user(login: str) → list[list[str]]

Функция выполняет выборку записей со статусом «Избранное» для пользователя

Параметры

`login (str)` – Логин пользователя

SQL-запрос (self.query_select_for_filter_all):

```
SELECT * FROM stateFavorites WHERE login = ?;
```

Результат

Результаты выборки по фильтру

Тип результата

`list[list[str]]`

3.2.6 Модель представления view_filter - класс ViewFilter

```
class database.sqlite3_interface.views.view_filter.ViewFilter(name_db: str)
```

Базовые классы: `View`

Класс является моделью представления filter.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания представления

`self.query_select_for_filter_all (str)`

Запрос выборки записей представления по критериям тэг, год и язык

`self.query_select_for_filter_tag_year (str)`

Запрос выборки записей представления по критериям тэг и год

`self.query_select_for_filter_tag_language (str)`

Запрос выборки записей представления по критериям тэг и язык

`self.query_select_for_filter_year_language (str)`

Запрос выборки записей представления по критериям язык и год

`self.query_select_for_filter_tag (str)`

Запрос выборки записей представления по критерию тэг

`self.query_select_for_filter_language (str)`

Запрос выборки записей представления по критерию языка

`self.query_select_for_filter_year (str)`

Запрос выборки записей представления по критерию год

SQL-запрос создания представления filter (self.create_query):

```
CREATE VIEW IF NOT EXISTS filter AS
SELECT book.id_book, book.title, book.path_book, book.path_image,
about.information, language.name AS language, year.year,
category.name, tag.tag_name FROM book
INNER JOIN about ON about.id_book = book.id_about
INNER JOIN language ON language.id_language = book.id_language
INNER JOIN category ON category.id_category = book.id_category
INNER JOIN year ON year.id_year = book.id_year
INNER JOIN list_tags ON list_tags.id_book = book.id_book
INNER JOIN tag ON list_tags.id_tag = tag.id_tag;
```

`select_for_filter_all(tag: str, year: int, language: str) → list[list[str]]`

Функция выполняет фильтр по критериям тэг, год и язык

Параметры

- `tag (str)` – Заданный тэг
- `year (int)` – Заданный год
- `language (str)` – Заданный язык

SQL-запрос (`self.query_select_for_filter_all`):

```
SELECT * FROM filter WHERE tag_name LIKE ?
AND year LIKE ? AND language LIKE ?;
```

Результат

Результаты выборки по фильтру

Тип результата

`list[list[str]]`

`select_for_filter_language(language: str) → list[list[str]]`

Функция выполняет фильтр по критерию языка

Параметры

`language (str)` – Заданный язык

SQL-запрос (`self.query_select_for_filter_language`):

```
SELECT * FROM filter WHERE language LIKE ?;
```

Результат

Результаты выборки по фильтру

Тип результата

`list[list[str]]`

`select_for_filter_tag(tag: str) → list[list[str]]`

Функция выполняет фильтр по критерию тэга

Параметры

`tag (str)` – Заданный тэг

SQL-запрос (`self.query_select_for_filter_tag`):

```
SELECT * FROM filter WHERE tag_name LIKE ?;
```

Результат

Результаты выборки по фильтру

Тип результата

list[[list[str]]]

`select_for_filter_tag_language(tag: str, language: str) → list[list[str]]`

Функция выполняет фильтр по критериям тэг и язык

Параметры

- `tag (str)` – Заданный тэг
- `language (str)` – Заданный язык

SQL-запрос (`self.query_select_for_filter_tag_language`):

```
SELECT * FROM filter WHERE tag_name LIKE ?
AND language LIKE ?;
```

Результат

Результаты выборки по фильтру

Тип результата

list[[list[str]]]

`select_for_filter_tag_year(tag: str, year: int) → list[list[str]]`

Функция выполняет фильтр по критериям тэг и год

Параметры

- `tag (str)` – Заданный тэг
- `year (int)` – Заданный год

SQL-запрос (`self.query_select_for_filter_tag_year`):

```
SELECT * FROM filter WHERE tag_name LIKE ?
AND year LIKE ?;
```

Результат

Результаты выборки по фильтру

Тип результата

list[[list[str]]]

`select_for_filter_year(year: int) → list[list[str]]`

Функция выполняет фильтр по критерию год

Параметры

- `year (int)` – Заданный год

SQL-запрос (`self.query_select_for_filter_year`):

```
SELECT * FROM filter WHERE year LIKE ?;
```

Результат

Результаты выборки по фильтру

Тип результата

list[[list[str]]]

`select_for_filter_year_language(language: str, year: int) → list[list[str]]`

Функция выполняет фильтр по критериям язык и год

Параметры

- `language (str)` – Заданный язык
- `year (int)` – Заданный год

SQL-запрос (self.query_select_for_filter_year_language):

```
SELECT * FROM search WHERE year LIKE ?
AND language LIKE ?;
```

Результат

Результаты выборки по фильтру

Тип результата

list[[list[str]]]

3.2.7 Модель представления `view_in_read` - класс `ViewInRead`

`class database.sqlite3_interface.views.view_in_read.ViewInRead(name_db: str)`

Базовые классы: `View`

Класс является моделью представления `state_read`.

Параметры

`name_db (str)` – Имя базы данных

Атрибуты:

`self.create_query (str)`

Запрос создания представления

`self.query_select_all (str)`

Запрос выборки всех записей со статусом «На чтении»

`self.query_select_for_user (str)`

Запрос выборки всех записей представления со статусом «На чтении» для пользователя

SQL-запрос создания представления `state_read` (`self.create_query`):

```
CREATE VIEW IF NOT EXISTS state_read AS
SELECT book.id_book, book.title, book.path_book,
book.path_image, about.information, language.name AS language,
year.year, category.name, user.id_user,
```

(continues on next page)

(продолжение с предыдущей страницы)

```

user.login, state.state_name      FROM book
INNER JOIN about ON about.id_book = book.id_about
INNER JOIN language ON language.id_language = book.id_language
INNER JOIN category ON category.id_category = book.id_category
INNER JOIN year ON year.id_year = book.id_year
INNER JOIN list_states ON list_states.id_book = book.id_book
INNER JOIN state ON list_states.id_state = state.id_state
INNER JOIN user ON list_states.id_user = user.id_user
WHERE state.state_name = 'На чтении';

```

SQL-запрос (self.query_select_all):

```
SELECT * FROM state_read;
```

select_for_user(login: str) → list[list[str]]

Функция выполняет выборку записей со статусом «На чтении» для пользователя

Параметры

title (str) – Заданное название книги

SQL-запрос (self.query_select_for_user):

```
SELECT * FROM state_read WHERE login = ?;
```

Результат

Результаты выборки

Тип результата

list[list[str]]

3.2.8 Модель представления view_joornals - класс ViewJoornals

class database.sqlite3_interface.views.view_joornals.ViewJoornals(name_db: str)Базовые классы: *View*

Класс является моделью представления joornals_category.

Параметры

name_db (str) – Имя базы данных

Атрибуты:**self.create_query (str)**

Запрос создания представления

self.query_select_all (str)

Запрос выборки всех записей категории «Журналы»

SQL-запрос создания представления joornals_category (self.create_query):

```

CREATE VIEW IF NOT EXISTS joornals_category AS
SELECT book.id_book, book.title, book.path_book,
book.path_image, about.information, language.name AS language,

```

(continues on next page)

(продолжение с предыдущей страницы)

```
year.year, category.name FROM book
INNER JOIN about ON about.id_book = book.id_about
INNER JOIN language ON language.id_language = book.id_language
INNER JOIN category ON category.id_category = book.id_category
INNER JOIN year ON year.id_year = book.id_year
WHERE category.name = 'Журналы'
```

SQL-запрос (self.query_select_all):

```
SELECT * FROM joornals_category;
```

3.2.9 Модель представления view_search - класс ViewSearch

```
class database.sqlite3_interface.views.view_search.ViewSearch(name_db: str)
```

Базовые классы: *View*

Класс является моделью представления search.

Параметры

name_db (*str*) – Имя базы данных

Атрибуты:

self.create_query (str)

Запрос создания представления

self.query_select_all (str)

Запрос выборки всех записей из представления search

self.query_select_on_title (str)

Запрос выборки записей с заданным названием книги

SQL-запрос создания представления search (self.create_query):

```
CREATE VIEW IF NOT EXISTS search AS
SELECT book.id_book, book.title, book.path_book,
book.path_image, about.information, language.name AS language,
year.year, category.name FROM book
INNER JOIN about ON about.id_book = book.id_about
INNER JOIN language ON language.id_language = book.id_language
INNER JOIN category ON category.id_category = book.id_category
INNER JOIN year ON year.id_year = book.id_year;
```

SQL-запрос (self.query_select_all):

```
SELECT * FROM search;
```

select_on_title(title: str) → list[list[str]]

Функция выполняет выборку записей по заданному названию книги

Параметры

title (*str*) – Заданное название книги

SQL-запрос (self.query_select_on_title):

```
SELECT * FROM search WHERE title LIKE ?;
```

Результат

Результаты поиска

Тип результата

list[[list[str]]]

3.3 Класс CreateDataBase

```
class database.sqlite3_interface.create_db.CreateDataBase(name_db: str)
```

Базовые классы: object

Класс предназначен для создания базы данных.

Параметры

name_db (str) – Имя базы данных

Атрибуты:**self.about (str)**

Объект класса About

self.author (str)

Объект класса Author

self.book (str)

Объект класса Book

self.category (str)

Объект класса Category

self.language (str)

Объект класса Language

self.list_authors (str)

Объект класса ListAuthors

self.list_notes (str)

Объект класса ListNotes

self.list_states (str)

Объект класса ListStates

self.list_tags (str)

Объект класса ListTags

self.note (str)

Объект класса Note

self.state (str)

Объект класса State

self.tag (str)

Объект класса Tag

self.year (str)

Объект класса Year

```
self.user (str)
    Объект класса User

self.view_articles (str)
    Объект класса ViewArticles

self.view_books (str)
    Объект класса ViewBooks

self.view_end_read (str)
    Объект класса ViewEndRead

self.viewFavorites (str)
    Объект класса ViewFavorites

self.view_in_read (str)
    Объект класса ViewInRead

self.view_journals (str)
    Объект класса ViewJournals

self.view_search (str)
    Объект класса ViewSearch

self.view_filter (str)
    Объект класса ViewFilter

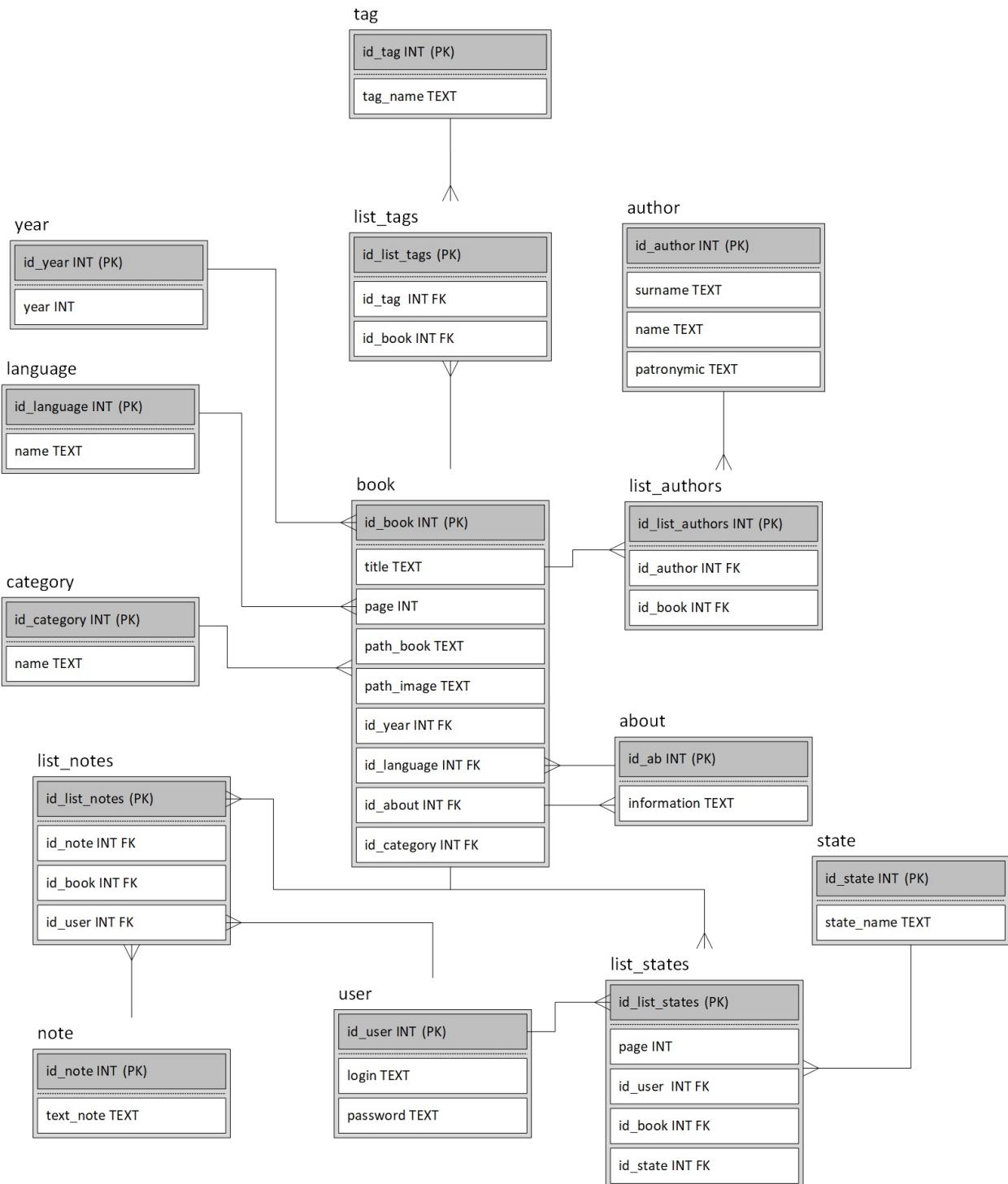
create_tables() → None
    Функция создания таблиц базы данных

create_views() → None
    Функция создания представлений базы данных

insert_data_to_db() → None
    Функция добавления тестовых данных в таблицу

insert_minimum_data_to_db() → None
    Функция добавления минимальных данных в таблицу: 1. Тэги 2. Язык 3. Год 4. Категория
    5. Статус 6. Пользователь
```

3.4 Физическая модель базы данных



Книжная библиотека

4.1 Компоненты приложения

4.1.1 Компонент `add_to_database`

Компонент предназначен для добавления книги.

```
blueprints.add_to_database.add(login: str) → render_template
```

Переход на страницу добавления книги.

Параметры

`login (string)` – Имя пользователя

Результаты выполнения условий:

```
if current_app.config  
    Переход на страницу add_file.html  
else  
    Переход на страницу index.html
```

Результат

`render_template: Страница html`

Возвращаемые html страницы:

`add_file.html`

Возвращает страницу добавления книги если сессия работы с библиотекой не завершена

`index.html`

Возвращает страницу авторизации если сессия работы с библиотекой прервана

`blueprints.add_to_database.add_file() → None`

Запрос добавления книги в базу данных. В данной функции выполняется многоуровневая проверка каждой переменной с выполнением запросов добавления согласно связям между таблицами базами данных. Сначала выполняется добавление книги с необходимой для нее информацией, затем выполняется добавление списков авторов и тэгов с заранне подготовленными данными для добавления.

Переменные:

title (str)

Название новой книги

page (str)

Имя нового автора

path_book (str)

Отчество нового автора

path_image (str)

Отчество нового автора

category (str)

Отчество нового автора

language (str)

Отчество нового автора

year (str)

Отчество нового автора

authors (list[str])

Список авторов для новой книги

tags (list[str])

Список тэгов для новой книги

Результат

`str`: Возвращает пустую строку

`blueprints.add_to_database.add_new_author() → None`

Запрос добавления нового автора в базу данных

Переменные:

surname (str)

Фамилия нового автора

name (str)

Имя нового автора

patronymic (str)

Отчество нового автора

Результаты выполнения условий:

`if not current_app.config`: выполняется запрос `insert`

`else`: ничего не выполняется

Результат

`str`: Возвращает пустую строку

Тип результата

str

`blueprints.add_to_database.add_new_tag() → None`

Запрос добавления нового тэга в базу данных

Переменные:**tag (str)**

Новый тэг

Результаты выполнения условий:**if not current_app.config**

выполняется запрос insert

else

ничего не выполняется

Результат

str: Возвращает пустую строку

`blueprints.add_to_database.change_add_author() → None`

Запрос добавления автора/авторов к книге. В данной функции выполняется многоуровневая проверка каждой переменной, формируется список для добавления и затем выполняется запрос insert.

Переменные:**id_book (str)**

Фамилия нового автора

authors (str)

Список авторов

Результат

str: Возвращает пустую строку

`blueprints.add_to_database.change_add_tag()`

Запрос добавления тэга/тэгов к книге. В данной функции выполняется многоуровневая проверка каждой переменной, формируется список для добавления и затем выполняется запрос insert.

Переменные:**id_book (str)**

Фамилия нового автора

tags (str)

Список тэгов

Результат

str: Возвращает пустую строку

4.1.2 Компонент auth

Компонент предназначен для авторизации.

`blueprints.auth.exit() → render_template`

Выход из личного кабинета

Переменные:

`current_app.config[“USER_LOGIN”] (str)`
имя пользователя

Результат

`render_template: Страница html`

Возвращаемые html страницы:

`index.html`

Возвращает страницу авторизации если сессия работы с библиотекой прервана

`blueprints.auth.index() → render_template`

Переход на страницу авторизации

Результат

`render_template: Страница html`

Возвращаемые html страницы:

`index.html`

Возвращает страницу авторизации если сессия работы с библиотекой прервана

`blueprints.auth.main() → render_template`

Переход на главную страницу

Переменные:

`current_app.config[“USER_LOGIN”] (str)`
имя пользователя

`login_enter (str)`

Введенный логин

`password_enter (str)`

Введенный пароль

`password_user (str)`

Пароль пользователя из базы данных

Параметры для `render_template`:

`login (str)`

Логин пользователя

`style (str)`

Стиль приложения

`tags (Tag)`

Модель таблицы базы данных tag

language (Language)

Модель таблицы базы данных language

year (Year)

Модель таблицы базы данных year

Результаты выполнения условий:

```
:if login_enter and password_enter and current_app.config[„USER“].select_password_on_login(login_enter):
    Выборка пароля из базы данных для пользователя
```

if password_user

Допуск к следующей проверки

if password_user == password_enter

Сохранение логина в переменной приложения

if current_app.config[„USER_LOGIN“]

Переход на страницу main.html

else

Переход на страницу index.html

Результат

render_template: Страница html

Возвращаемые html страницы:**index.html**

Возвращает страницу авторизации если сессия работы с библиотекой прервана

4.1.3 Компонент book_read

Компонент предназначен для чтения книги.

`blueprints.book_read.add_note() → str`

Запрос добавления заметки

Переменные:**text_note (str)**

Новая заметка

id_book (int)

id книги

id_user (int)

id пользователя

Результаты выполнения условий:

```
if id_last_note and id_book and id_user
    добавляется новая заметка для книги
```

else

ничего не выполняется, одна из переменных со значением null

Результат

str: Возвращает пустую строку

`blueprints.book_read.change_page_marker() → str`

Запрос изменения страницы закладки

Переменные:

`new_page (str)`
Новая страница закладки

`id_book (int)`
id книги

`id_user (int)`
id пользователя

Результаты выполнения условий:

```
if id_book and id_user and new_page
    выполняется запрос update для таблицы list_states
else
    ничего не выполняется
```

Результат

`str`: Возвращает пустую строку

`blueprints.book_read.open_book(login: str, id_book: int) → render_template`

Переход на страницу открытия книги

Параметры

- `login (string)` – Имя пользователя
- `id_book (int)` – id книги

Результаты выполнения условий:

```
if check_book_on_user
    выполняется запрос update для таблицы list_states
else
    выполняется запрос insert для таблицы list_states
```

Результат

`render_template`: Страница html

Возвращаемые html страницы:

open_book.html
Возвращает страницу чтения книги

index.html
Возвращает страницу авторизации если сессия работы с библиотекой прервана

4.1.4 Компонент `change_book_information`

Компонент предназначен для изменения информации о книге.

```
blueprints.change_book_information.change_book(login: str, id_book: int) → render_template
```

Переход на страницу изменения книги

Параметры

- `login (string)` – Имя пользователя
- `id_book (int)` – id книги

Результаты выполнения условий:

```
if current_app.config
    Переход на страницу change_book.html
else
    Переход на страницу index.html
```

Результат

`render_template: Страница html`

Возвращаемые html страницы:

change_book.html
Возвращает страницу изменения книги

index.html
Возвращает страницу авторизации если сессия работы с библиотекой прервана

```
blueprints.change_book_information.change_category() → str
```

Запрос изменения категории книги

Переменные:

category (str)
Новая категория книги

id_book (int)
id книги

Результаты выполнения условий:

```
if category and id_book
    выполняется запрос update для выбранной книги
else
    ничего не выполняется
```

Результат

`str: Возвращает пустую строку`

```
blueprints.change_book_information.change_description() → str
```

Запрос изменения описания книги

Переменные:

description (str)

Новое описание книги

id_book (int)

id книги

Результаты выполнения условий:

if language and id_book

выполняется запрос update для выбранной книги

else

ничего не выполняется

Результат

str: Возвращает пустую строку

`blueprints.change_book_information.change_language() → str`

Запрос изменения языка книги

Переменные:

language (str)

Новый язык книги

id_book (int)

id книги

Результаты выполнения условий:

if language and id_book

выполняется запрос update для выбранной книги

else

ничего не выполняется

Результат

str: Возвращает пустую строку

`blueprints.change_book_information.change_page() → str`

Запрос изменения количества страниц книги

Переменные:

page (str)

Новое количество страниц книги

id_book (int)

id книги

Результаты выполнения условий:

if page and id_book

выполняется запрос update для выбранной книги

else

ничего не выполняется

Результат

str: Возвращает пустую строку

```
blueprints.change_book_information.change_path_book() → str
```

Запрос изменения пути к файлу книги

Переменные:

path_book (str)

Новый путь к файлу книги

id_book (int)

id книги

Результаты выполнения условий:

if path_book and id_book

выполняется запрос update для выбранной книги

else

ничего не выполняется

Результат

str: Возвращает пустую строку

```
blueprints.change_book_information.change_path_image() → str
```

Запрос изменения пути к изображению книги

Переменные:

path_book (str)

Новый путь к изображению книги

id_book (int)

id книги

Conditions:

if path_image and id_book

выполняется запрос update для выбранной книги

else

ничего не выполняется

Результат

str: Возвращает пустую строку

```
blueprints.change_book_information.change_title() → str
```

Запрос на изменение названия книги

Переменные:

title (str)

Новое название книги

id_book (int)

id книги

Результаты выполнения условий:

if title and id_book

выполняется запрос update для выбранной книги

else

ничего не выполняется

Результат

str: Возвращает пустую строку

blueprints.change_book_information.change_year() → str

Запрос изменения года издания книги

Переменные:

year (str)

Новый год издания книги

id_book (int)

id книги

Результаты выполнения условий:

if language and id_book

выполняется запрос update для выбранной книги

else

ничего не выполняется

Результат

str: Возвращает пустую строку

4.1.5 Компонент change_state

Компонент предназначен для изменения статуса книги.

blueprints.change_state.change_end_read(login: str, id_book: int) → str

Запрос для добавление/изменения книги в прочитанное/на прочитанное

Параметры

- **login (str)** – Имя пользователя

- **id_book (int)** – id книги

Результаты выполнения условий:

if check_book_on_user

выполняется запрос update для таблицы list_states

else

выполняется запрос insert для таблицы list_states

Результат

str: Возвращает пустую строку

blueprints.change_state.change_in_read(login: str, id_book: int) → str

Запрос для добавление/изменения книги на чтение

Параметры

- **login (str)** – Имя пользователя

- **id_book (int)** – id книги

Результаты выполнения условий:

```
if check_book_on_user
    выполняется запрос update для таблицы list_states
else
    выполняется запрос insert для таблицы list_states
```

Результат

str: Возвращает пустую строку

`blueprints.change_state.change_on_favorite(login: str, id_book: int) → str`

Запрос для добавление/изменения книги в избранное

Параметры

- `login (str)` – Имя пользователя
- `id_book (int)` – id книги

Результаты выполнения условий:

```
if check_book_on_user
    выполняется запрос update для таблицы list_states
else
    выполняется запрос insert для таблицы list_states
```

Результат

str: Возвращает пустую строку

4.1.6 Компонент `function`

Компонент содержит вспомогательные функции.

`blueprints.function.check_book_on_user(books_by_user: list[str], book_in_check: str) → bool`

Эта функция проверяет наличие книги у пользователя

Параметры

- `books_by_user (list [str])` – Список книг автора
- `book_in_check (str)` – Книга сравниваемая с книгами пользователя

Variable:

`param check (bool)`

Переменная проверки книги, `check = False`, если книга имеется у пользователя
`check = True`

Результат

`bool: True - книга имеется у пользователя, False - книга отсутствует у пользователя`

`blueprints.function.formation_authors(list_words: list[str]) → list[str]`

Эта функция формирует список авторов из списка слов ([Фамилия имя отчество] -> [Фамилия, Имя, Отчество])

Параметры

list_words (*list [str]*) – Список слов

Результат

list[str]: Список авторов

blueprints.function.formation_tags(*list_words: list[str]*) → list[str]

Эта функция формирует список тэгов из списка слов ([Тэг Тэг] -> [Тэг, Тэг])

Параметры

list_words (*list [str]*) – Список слов

Результат

list[str]: Список тэгов

blueprints.function.parser_html_ajax(*id_container: int*) → findall

Эта функция парсинга данных принятых с аjax в виде html в список строк

Параметры

id_container (int) – id контейнера данных html

Functions:

function re.findall()

- Регулярное выражение с правилом „(?=>)[^<>]+(?=</)“, выбирающем текст внутри тегов html

Результат

re.findall: Список строк сформированный с помощью регулярных выражений

blueprints.function.parser_list_in_list(*list_in_list: list[list[list[str]]]*) → list[list[str]]

Эта функция уменьшает на один уровень вложенности списка :param list_in_list: Многоуровневый список(больше трех уровней) :type list_in_list: list[list[str]]

Результат

list [list[str]]: Список с уменьшенным количеством уровней и уникальными строками

4.1.7 Компонент search_filter

Компонент предназначен для формирования страниц найденных книг.

blueprints.search_filter.filter(*login: str, page: int*) → render_template

Эта функция возвращает все найденные книжные издания по фильтру

Параметры

- login (*string*) – Имя пользователя
- page (*int*) – Страница

Переменные:

year (int)

Заданный год

tag (str)

Заданный тэг

language (str)
Заданный язык

Параметры для render_template:

books (ViewFilter)

Модель представления базы данных filter

page (int)

Страница на которой находится пользователь (по умолчанию 1)

login (str)

Логин пользователя

style (str)

Стиль приложения

tags (Tag)

Модель таблицы базы данных tag

authors (str)

Модель таблицы базы данных authors

language (Language)

Модель таблицы базы данных language

year (Year)

Модель таблицы базы данных year

category (str)

Категория книг, необходимая для формирования перехода между страницами

state (State)

Модель таблицы базы данных state

id_user (int)

id пользователя

Результат

render_template: Страница html

Возвращаемые html страницы:

:books.html : Возвращает страницу с книгами если сессия работы с библиотекой не завершена

main.html

Возвращает главную страницу если у пользователя отсутствуют книги с выбранным статусом

index.html

Возвращает страницу авторизации если сессия работы с библиотекой прервана

Результаты выполнения условий подготовки фильтра (аналогично для двух других блоков условий):

if page == 1

FILTER_YEAR, FILTER_LANGUAGE и FILTER_TAG сбрасываются

if [„FILTER_YEAR“] == None

Присвоение переменной значения

elif („year“) and („year“) != [„FILTER_YEAR“]

Присвоение переменной значения

Результаты выполнения условий фильтра:

if year and language and tag

Возвращает страницу с книгами по трем критериям (год, язык и тэг)

elif tag and year

Возвращает страницу с книгами по двум критериям (тэг и год)

elif tag and language

Возвращает страницу с книгами по двум критериям (тэг и язык)

elif language and year

Возвращает страницу с книгами по двум критериям (язык и год)

elif tag

Возвращает страницу с книгами по одному критерию (тэг)

elif language

Возвращает страницу с книгами по одному критерию (язык)

elif year

Возвращает страницу с книгами по одному критерию (год)

`blueprints.search_filter.search(login: str, page: int) → render_template`

Эта функция возвращает все найденные книжные издания

Параметры

- `login (string)` – Имя пользователя
- `page (int)` – Страница

Параметры для render_template:

books (ViewSearch)

Модель представления базы данных search

page (int)

Страница на которой находится пользователь (по умолчанию 1)

login (str)

Логин пользователя

style (str)

Стиль приложения

tags (Tag)

Модель таблицы базы данных tag

authors (str)

Модель таблицы базы данных authors

language (Language)

Модель таблицы базы данных language

year (Year)

Модель таблицы базы данных year

category (str)

Категория книг, необходимая для формирования перехода между страницами

state (State)

Модель таблицы базы данных state

id_user (int)
id пользователя

Результат

render_template: Страница html

Возвращаемые html страницы:**books.html**

Возвращает страницу с книгами если сессия работы с библиотекой не завершена

main.html

Возвращает главную страницу если у пользователя отсутствуют книги с выбранным статусом

index.html

Возвращает страницу авторизации если сессия работы с библиотекой прервана

4.1.8 Компонент select_books

Компонент предназначен для формирования страниц книг с различными категориями («Книги», «Журналы», «Статьи»).

`blueprints.select_books.article(login: str, page: int) → render_template`

Эта функция возвращает страницу со статьями

Параметры

- **login (string)** – Имя пользователя
- **page (int)** – Страница

Параметры для render_template:**books (ViewArticles)**

Модель представления базы данных view_articles

page (int)

Страница на которой находится пользователь (по умолчанию 1)

login (str)

Логин пользователя

style (str)

Стиль приложения

tags (Tag)

Модель таблицы базы данных tag

authors (str)

Модель таблицы базы данных authors

language (Language)

Модель таблицы базы данных language

year (Year)

Модель таблицы базы данных year

category (str)

Категория книг, необходимая для формирования перехода между страницами

state (State)

Модель таблицы базы данных state

id_user (int)

id пользователя

Результат

render_template: Страница html

Возвращаемые html страницы:

books.html

Возвращает страницу с книгами если сессия работы с библиотекой не завершена

main.html

Возвращает главную страницу если у пользователя отсутствуют книги с выбранной категорией

index.html

Возвращает страницу авторизации если сессия работы с библиотекой прервана

`blueprints.select_books.books(login: str, page: int) → render_template`

Эта функция возвращает страницы с книгами

Параметры

- `login (string)` – Имя пользователя
- `page (int)` – Страница

Параметры для render_template:

books (ViewBooks)

Модель представления базы данных view_books

page (int)

Страница на которой находится пользователь (по умолчанию 1)

login (str)

Логин пользователя

style (str)

Стиль приложения

tags (Tag)

Модель таблицы базы данных tag

authors (str)

Модель таблицы базы данных authors

language (Language)

Модель таблицы базы данных language

year (Year)

Модель таблицы базы данных year

category (str)

Категория книг, необходимая для формирования перехода между страницами

state (State)

Модель таблицы базы данных state

id_user (int)

id пользователя

Результат

render_template: Страница html

Возвращаемые html страницы:**books.html**

Возвращает страницу с книгами если сессия работы с библиотекой не завершена

main.html

Возвращает главную страницу если у пользователя отсутствуют книги с выбранной категорией

index.html

Возвращает страницу авторизации если сессия работы с библиотекой прервана

`blueprints.select_books.joornal(login: str, page: int) → render_template`

Эта функция возвращает страницы журналов

Параметры

- `login (string)` – Имя пользователя
- `page (int)` – Страница

Параметры для render_template:**books (ViewJoornals)**

Модель представления базы данных view_journals

page (int)

Страница на которой находится пользователь (по умолчанию 1)

login (str)

Логин пользователя

style (str)

Стиль приложения

tags (Tag)

Модель таблицы базы данных tag

authors (str)

Модель таблицы базы данных authors

language (Language)

Модель таблицы базы данных language

year (Year)

Модель таблицы базы данных year

category (str)

Категория книг, необходимая для формирования перехода между страницами

state (State)

Модель таблицы базы данных state

id_user (int)
id пользователя

Результат

render_template: Страница html

Возвращаемые html страницы:

books.html

Возвращает страницу с книгами если сессия работы с библиотекой не завершена

main.html

Возвращает главную страницу если у пользователя отсутствуют книги с выбранной категорией

index.html

Возвращает страницу авторизации если сессия работы с библиотекой прервана

4.1.9 Компонент select_books_state

Компонент предназначен для формирования страниц книг с различными статусами («На чтении», «Избранное», «Прочитанное»).

`blueprints.select_books_state.end_read(login: str, page: int) → render_template`

Эта функция возвращает все категории книжных изданий со статусом «Прочитанное»

Параметры

- `login (string)` – Имя пользователя
- `page (int)` – Страница

Параметры для render_template:

books (ViewEndRead)

Модель представления базы данных view_end_read

page (int)

Страница на которой находится пользователь (по умолчанию 1)

login (str)

Логин пользователя

style (str)

Стиль приложения

tags (Tag)

Модель таблицы базы данных tag

authors (str)

Модель таблицы базы данных authors

language (Language)

Модель таблицы базы данных language

year (Year)

Модель таблицы базы данных year

category (str)

Категория книг, необходимая для формирования перехода между страницами

state (State)

Модель таблицы базы данных state

id_user (int)

id пользователя

Результат

Страница html

Тип результата

render_template

Возвращаемые html страницы:**books.html**

Возвращает страницу с книгами если сессия работы с библиотекой не завершена

main.html

Возвращает главную страницу если у пользователя отсутствуют книги с выбранным статусом

index.html

Возвращает страницу авторизации если сессия работы с библиотекой прервана

`blueprints.select_books_state.favorites(login: str, page: int) → render_template`

Эта функция возвращает все категории книжных изданий со статусом «Избранное»

Параметры

- `login (string)` – Имя пользователя
- `page (int)` – Страница

Параметры для render_template:**books (ViewFavorites)**

Модель представления базы данных view_favorites

page (int)

Страница на которой находится пользователь (по умолчанию 1)

login (str)

Логин пользователя

style (str)

Стиль приложения

tags (Tag)

Модель таблицы базы данных tag

authors (str)

Модель таблицы базы данных authors

language (Language)

Модель таблицы базы данных language

year (Year)

Модель таблицы базы данных year

category (str)

Категория книг, необходимая для формирования перехода между страницами

state (State)

Модель таблицы базы данных state

id_user (int)

id пользователя

Результат

render_template: Страница html

Возвращаемые html страницы:

books.html

Возвращает страницу с книгами если сессия работы с библиотекой не завершена

main.html

Возвращает главную страницу если у пользователя отсутствуют книги с выбранным статусом

index.html

Возвращает страницу авторизации если сессия работы с библиотекой прервана

`blueprints.select_books_state.on_reading(login: str, page: int) → render_template`

Эта функция возвращает все категории книжных изданий со статусом «На чтении»

Параметры

- `login (string)` – Имя пользователя
- `page (int)` – Страница

Параметры для render_template:

books (ViewInRead)

Модель представления базы данных view_in_read

page (int)

Страница на которой находится пользователь (по умолчанию 1)

login (str)

Логин пользователя

style (str)

Стиль приложения

tags (Tag)

Модель таблицы базы данных tag

authors (str)

Модель таблицы базы данных authors

language (Language)

Модель таблицы базы данных language

year (Year)

Модель таблицы базы данных year

category (str)

Категория книг, необходимая для формирования перехода между страницами

state (State)

Модель таблицы базы данных state

id_user (int)

id пользователя

Результат

render_template: Страница html

Возвращаемые html страницы:

books.html

Возвращает страницу с книгами если сессия работы с библиотекой не завершена

main.html

Возвращает главную страницу если у пользователя отсутствуют книги с выбранным статусом

index.html

Возвращает страницу авторизации если сессия работы с библиотекой прервана

4.1.10 Компонент style

Компонент предназначен для изменения стиля сайта.

`blueprints.style.style1() → str`

Эта функция меняет стиль страницы

Результат

Изменение стиля

Тип результата

str

`blueprints.style.style2() → str`

Эта функция меняет стиль страницы

Результат

Изменение стиля

Тип результата

str

`blueprints.style.style3() → str`

Эта функция меняет стиль страницы

Результат

Изменение стиля

Тип результата

str

`blueprints.style.style4() → str`

Эта функция меняет стиль страницы

Результат

Изменение стиля

Тип результата

str

`blueprints.style.style5() → str`

Эта функция меняет стиль страницы

Результат

Изменение стиля

Тип результата

str

JavaScript код

5.1 add_file.js

```
/**  
 * Функция добавление нового элемента option в select со значением из другого  
 ↵option  
 * @function add_in_list  
 * @param {string} id элемента из которого копируются данные  
 * @param {string} id элемента куда копируются данные  
 */  
function add_in_list(id_from, id_to){  
    //создание элемента option  
    const newDiv = document.createElement("option");  
  
    //получения выбранного option из select  
    var value = $(id_from).text();  
    //помещение текста в option  
    const newContent = document.createTextNode(value);  
  
    //установка новому option свойство selected  
    newDiv.setAttribute('selected','selected')  
  
    //добавление текста в option  
    newDiv.appendChild(newContent);  
  
    //Получение select для добавления нового тега  
    const currentDiv = document.getElementById(id_to);  
    currentDiv.appendChild(newDiv);  
};
```

```
/** 
 * ajax запрос добавления нового тэга
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для добавления нового тэга
 * @param {string} tag - название нового тэга
 */
$(document).ready(function() {
    $('#new_tag').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/add_new_tag',
            data: {tag:$('#new_tag_book_id').val()},
            success: function() {
                alert('Тэг добавлен!');
            }
        });
    });
});
```

```
/** 
 * ajax запрос добавления нового автора
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для добавления нового автора
 * @param {string} surname - фамилия нового автора
 * @param {string} name - имя нового автора
 * @param {string} patronymic - отчество нового автора
 */
$(document).ready(function() {
    $('#btn_new_author').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/add_new_author',
            data: {surname:$('#surname_author_id').val(), name:$('#name_author_id').val(), patronymic:$('#patronymic_author_id').val()},
            success: function() {
                alert('Автор добавлен!');
            }
        });
    });
});
```

```
/** 
 * ajax запрос добавления новой книги
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для добавления новой книги
 * @param {html} tags - список тэгов для новой книги
 * @param {html} authors - список авторов для новой книги
 * @param {string} title - название новой книги
 * @param {number} page - количество страниц новой книги
 * @param {string} path_book - путь к файлу новой книги
 */

```

(continues on next page)

(продолжение с предыдущей страницы)

```

 * @param {string} path_image - путь к изображению новой книги
 * @param {string} category - категория новой книги
 * @param {string} language - язык новой книги
 * @param {string} year - год издания новой книги
 * @param {string} year - описание новой книги
 */
$(document).ready(function() {
    $('#btn_add_file').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/add_file',
            data: {tags:$('#selected_tags').html(), authors:$('#selected_
→authors_list').html(), title: $('#title_book').val(),
                page: $('#count_pages').val(), path_book: $('#path_
→file_book').val(), path_image: $('#path_image_book').val(),
                category: $('#category_book').val(), language: $('#_
→language_book').val(), year: $('#year_book').val(),
                description: $('#description_book_id').val()
            },
            success: function() {
                alert('Книга добавлена!');
            }
        });
    });
});

```

5.2 change_file.js

```

 /**
 * ajax запрос изменения названия книги
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для изменения названия книги
 * @param {string} title - новое название книги
 * @param {number} id_book - id книги
 */
$(document).ready(function() {
    $('#change_name_book').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_title',
            data: {title: $('#title_book').val(), id_book: id_book},
            success: function() {
                alert('Название изменено!');
            }
        });
    });
});

```

```
/***
*ajax запрос изменения количества страниц книги
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для изменения количества страниц
→ книгу
 * @param {number} page - новое количество страниц книги
 * @param {number} id_book - id книги
*/
$(document).ready(function() {
    $('#change_page_book').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_page',
            data: {page: $('#count_pages').val(), id_book: id_book},
            success: function() {
                alert('Количество страниц изменено!');
            }
        });
    });
});
```

```
/***
*ajax запрос изменения пути к файлу книги
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для изменения пути к файлу книги
 * @param {string} path_book - новый путь к файлу книги
 * @param {number} id_book - id книги
*/
$(document).ready(function() {
    $('#change_path_book').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_path_book',
            data: {path_book: $('#path_file_book').val(), id_book: id_
book},
            success: function() {
                alert('Путь к файлу книги изменен!');
            }
        });
    });
});
```

```
/***
*ajax запрос изменения пути к изображению книги
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для изменения пути к изображению
→ книгу
 * @param {string} path_image - новый путь к изображению книги
 * @param {number} id_book - id книги
*/
*/
```

(continues on next page)

(продолжение с предыдущей страницы)

```

$(document).ready(function() {
    $('#change_path_image_book').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_path_image',
            data: {path_image: $('#path_image_book').val(), id_book: id_
book},
            success: function() {
                alert('Путь к изображению книги изменен!');
            }
        });
    });
});

```

```

/**
*ajax запрос изменения категории книги
* @param {string} type - метод POST
* @param {string} url - маршрут flask для изменения категории книги
* @param {string} category - новая категория книги
* @param {number} id_book - id книги
*/
$(document).ready(function() {
    $('#change_category_book').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_category',
            data: {category: $('#category_book').val(), id_book: id_book}
        },
        success: function() {
            alert('Категория книги изменена!');
        }
    });
});

```

```

/**
*ajax запрос изменения языка книги
* @param {string} type - метод POST
* @param {string} url - маршрут flask для изменения языка книги
* @param {string} language - новый язык книги
* @param {number} id_book - id книги
*/
$(document).ready(function() {
    $('#change_language_book').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_language',
            data: {language: $('#language_book').val(), id_book: id_book}
        });
    });
});

```

(continues on next page)

(продолжение с предыдущей страницы)

```

        },
        success: function() {
            alert('Язык книги изменен!');
        }
    });
});
}
);

```

```

/**
*ajax запрос изменения года издания книги
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для изменения года издания книги
 * @param {number} year - новый год издания книги
 * @param {number} id_book - id книги
*/
$(document).ready(function() {
    $('#change_year_book').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_year',
            data: {year: $('#year_book').val(), id_book: id_book},
            success: function() {
                alert('Год издания книги изменен!');
            }
        });
    });
});

```

```

/**
*ajax запрос изменения описания книги
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для изменения описания книги
 * @param {string} description - новое описание книги
 * @param {number} id_book - id книги
*/
$(document).ready(function() {
    $('#btn_change_description').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_description',
            data: {description: $('#description_book_id').val(), id_
book: id_book},
            success: function() {
                alert('Описание книги изменено!');
            }
        });
    });
});

```

```
/*
*ajax запрос добавления автора/авторов к книге
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для добавления автора/авторов к книге
 * @param {html} authors - автор/авторы
 * @param {number} id_book - id книги
*/
$(document).ready(function() {
    $('#btn_add_list_authors').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_add_author',
            data: {authors: $('#selected_authors_list').html(), id_book:id_book},
            success: function() {
                alert('Автор/авторы добавлены к книге!');
            }
        });
    });
});
```

```
/*
*ajax запрос добавления тэга/тэгов к книге
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для добавления тэга/тэгов к книге
 * @param {html} tags - тэг/тэги
 * @param {number} id_book - id книги
*/
$(document).ready(function() {
    $('#change_list_tags').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_add_tag',
            data: {tags: $('#selected_tags').html(), id_book: id_book},
            success: function() {
                alert('Тэг/тэги добавлены к книге!');
            }
        });
    });
});
```

5.3 change_state.js

```
/*
* ajax запрос добавления книги в избранное
* @param {string} type - метод POST
* @param {string} url - маршрут flask для добавления книги в избранное
* @param {string} login - логин пользователя
* @param {string} id_book - id книги
*/
function change_state_favorite(login, id_book)
{
    $(document).ready(function() {
        $('#favorite'+id_book).click(function(event) {
            event.preventDefault();
            $.ajax({
                type: 'POST',
                url: '/' + login + '/change_on_favorite/' + id_book,
                success: function(){
                    alert('Книга добавлена в избранное');
                }
            });
        });
    });
}
```

```
/*
* ajax запрос добавления книги на чтение
* @param {string} type - метод POST
* @param {string} url - маршрут flask для добавления книги на чтение
* @param {string} login - логин пользователя
* @param {string} id_book - id книги
*/
function change_state_in_read(login, id_book)
{
    $(document).ready(function() {
        $('#in_read'+id_book).click(function(event) {
            event.preventDefault();
            $.ajax({
                type: 'POST',
                url: '/' + login + '/change_in_read/' + id_book,
                success: function(){
                    alert('Книга добавлена на чтение');
                }
            });
        });
    });
}
```

```
/*
* ajax запрос добавления книги в прочитанное
* @param {string} type - метод POST
* @param {string} url - маршрут flask для добавления книги в

```

(continues on next page)

(продолжение с предыдущей страницы)

```

→ прочитанное
    * @param {string} login - логин пользователя
    * @param {string} id_book - id книги
*/
function change_state_end_read(login, id_book)
{
    $(document).ready(function() {
        $('#end_read'+id_book).click(function(event) {
            event.preventDefault();
            $.ajax({
                type: 'POST',
                url: '/' + login + '/change_end_read/' + id_book,
                success: function(){
                    alert('Книга перенесена в прочитанное
→ ');
                }
            });
        });
    });
}

```

5.4 filter_scripts.js

```

/**
*Функция открытия и закрытие окна фильтра
    * @param {boolean} filter_state - состояния окна (false - закрыто, true - открыто)
*/
$(document).ready(function(){
    var filter_state = false;
    $("#button_filter_id").click(function(){
        if (!filter_state)
        {
            $(".button_filter").css('visibility', 'visible');
            $(".select_filter").css('visibility', 'visible');
            $("#filter_main_window").css('visibility', 'visible');
            filter_state = true;
        }
        else
        {
            $(".button_filter").css('visibility', 'hidden');
            $(".select_filter").css('visibility', 'hidden');
            $("#filter_main_window").css('visibility', 'hidden');
            filter_state = false;
        }
    });
});

```

5.5 main_scripts.js

```
/***
*Функция изменения расположение элементов при нажатии кнопки скрыть
*/
$(document).ready(function(){
    $("#button_hide_id").click(function(){
        $("#button_show_id").show();
        $("#button_show_id").css("visibility","visible");
        $(".menu_button").hide();
        $(this).hide();
        $(".main_window_library").css("position","absolute");
        $(".main_window_library").css("left","0px");
        $(".main_window_library").css("top","-155px");
        $("#menu_style").css("top","-5px");
        $("#button_style1").css("top","250px");
        $("#button_style2").css("top","250px");
        $("#button_style3").css("top","250px");
        $("#button_style4").css("top","250px");
        $("#button_style5").css("top","250px");
    });
});
```

```
/***
*Функция изменения расположение элементов при нажатии кнопки показать
*/
$(document).ready(function(){
    $("#button_show_id").click(function(){
        $("#button_hide_id").show();
        $(".menu_button").show();//++
        $(this).hide();//++
        $(".main_window_library").css("position","absolute");
        $(".main_window_library").css("left","0px");
        $(".main_window_library").css("top","0px");
        $("#menu_style").css("top","150px");
        $("#button_style1").css("top","250px");
        $("#button_style2").css("top","250px");
        $("#button_style3").css("top","250px");
        $("#button_style4").css("top","250px");
        $("#button_style5").css("top","250px");
    });
});
```

```
/***
*Функция динамичного появления и исчезновения кнопок для изменения стилей
*/
$(document).ready(function(){
    $(".button_style").hover(
        function(){
            $(".button_style_position1").css("visibility","visible");
            $(".button_style_position2").css("visibility","visible");
            $(".button_style_position3").css("visibility","visible");
        }
    );
});
```

(continues on next page)

(продолжение с предыдущей страницы)

```

        $(".button_style_position4").css("visibility","visible");
        $(".button_style_position5").css("visibility","visible");
    },
    function(){
        $(".button_style_position1").css("visibility","hidden");
        $(".button_style_position2").css("visibility","hidden");
        $(".button_style_position3").css("visibility","hidden");
        $(".button_style_position4").css("visibility","hidden");
        $(".button_style_position5").css("visibility","hidden");
    });
});

```

```

/*
*Функция передвижения меню пользователя в процессе скроллинга
*/
window.addEventListener('scroll', function() {
    $(document).ready(function(){
        var elemValueScroll = 320;
        var valueScroll = String (window.pageYOffset+elemValueScroll);
        var setValueScroll = valueScroll + "px";
        $(".user_menu").css("top",setValueScroll);
    });
});

```

5.6 open_book.js

```

/*
*Функция открытия и закрытие окна закладки
 * @param {boolean} marker_state - состояния окна (false - закрыто, true - открыто)
*/
$(document).ready(function(){
    var marker_state=false;
    $("#btn_show_marker").click(function(){
        if (!marker_state)
        {
            $('.container_marker').css('visibility','visible');
            marker_state = true;
        }
        else
        {
            $('.container_marker').css('visibility','hidden');
            marker_state = false;
        }
    });
});

```

```

/*
*Функция открытия и закрытие окна заметок

```

(continues on next page)

(продолжение с предыдущей страницы)

```

    * @param {boolean} notes_state - состояния окна (false - закрыто, true
    ↵ - открыто)
*/
$(document).ready(function(){
    var notes_state=false;
    $("#show_notes").click(function(){
        if (!notes_state)
        {
            $('.container_notes').css('visibility','visible');
            notes_state = true;
        }
        else
        {
            $('.container_notes').css('visibility','hidden');
            notes_state = false;
        }
    });
});

```

```

/**
*Функция отображения книги во весь экран
*/
$(document).ready(function(){
    $("#scale_size").click(function(){
        $("#menu_book_id").css("top","-1940px");
        $("#book_window").css("left","0px");
        $("#book_window").css("top","-905px");
        $("#book_window").css("width","2220px");
        $("#book_window").css("height","888px");
        $("#frame_book").css("height","1888px");
        $("#frame_book").css("width","2220px");
    });
});

```

```

/**
*Функция свернуть окно книги
*/
$(document).ready(function(){
    $("#return_size").click(function(){
        $("#menu_book_id").css("top","-770px");
        $("#book_window").css("left","375px");
        $("#book_window").css("top","-585px");
        $("#book_window").css("width","1150px");
        $("#book_window").css("height","888px");
        $("#frame_book").css("height","720px");
        $("#frame_book").css("width","1150px");
    });
});

```

```

/**
*ajax запрос изменения страницы закладки

```

(continues on next page)

(продолжение с предыдущей страницы)

```

/* @param {string} type - метод POST
 * @param {string} url - маршрут flask для изменения страницы закладки
 * @param {string} new_page - новый страница закладки
 * @param {number} id_book - id книги
*/
$(document).ready(function() {
    $('#btn_change_marker').click(function(event) {
        event.preventDefault();
        $.ajax({
            type: 'POST',
            url: '/change_page_marker',
            data: {new_page: $('#value_marker').val(), id_book: id_book},
            success: function() {
                alert('Страница закладки изменена!');
            }
        });
    });
});

```

```

/**
* ajax запрос добавления заметки
* @param {string} type - метод POST
* @param {string} url - маршрут flask для изменения страницы закладки
* @param {string} text_note - текст новой заметки
* @param {number} id_book - id книги
*/
$(document).ready(function() {
    $('#add_note_button').click(function(event) {
        $.ajax({
            type: 'POST',
            url: '/add_note',
            data: {text_note: $('#input_field_note').val(), id_book: id_
book},
            success: function() {
                alert('Заметка добавлена!');
            }
        });
    });
});

```

5.7 shift_style.js

```

/**
* ajax запрос для стиля 1
* @param {string} type - метод POST
* @param {string} url - маршрут flask для изменения стиля сайта
*/
$(document).ready(function() {
    $('#button_style1').click(function() {

```

(continues on next page)

(продолжение с предыдущей страницы)

```
$.ajax({
    type: 'POST',
    url: '/style1'
});
```

```
/** 
 * ajax запрос для стиля 2
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для изменения стиля сайта
*/
$(document).ready(function() {
    $('#button_style2').click(function() {
        $.ajax({
            type: 'POST',
            url: '/style2'
        });
    });
});
```

```
/** 
 * ajax запрос для стиля 3
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для изменения стиля сайта
*/
$(document).ready(function() {
    $('#button_style3').click(function() {
        $.ajax({
            type: 'POST',
            url: '/style3'
        });
    });
});
```

```
/** 
 * ajax запрос для стиля 4
 * @param {string} type - метод POST
 * @param {string} url - маршрут flask для изменения стиля сайта
*/
$(document).ready(function() {
    $('#button_style4').click(function() {
        $.ajax({
            type: 'POST',
            url: '/style4'
        });
    });
});
```

```
/** 
 * ajax запрос для стиля 5
```

(continues on next page)

(продолжение с предыдущей страницы)

```

    * @param {string} type - метод POST
    * @param {string} url - маршрут flask для изменения стиля сайта
*/
$(document).ready(function() {
    $('#button_style5').click(function() {
        $.ajax({
            type: 'POST',
            url: '/style5'
        });
    });
});

```

```

/**
 * Функция изменения задания стиля 1
 * @function style1
 * Меняет стиль некоторых элементов сайта.
*/
function style1(){
    // Класс кнопок low_button (все кнопки кроме верхник и кнопок меню
    // пользователя)
    $(".low_button").css("color","rgb(22, 22, 22)");
    $(".low_button").css("background-color","rgb(255,174,66)");
    $(".low_button").css("box-shadow","0px 0px 0px 0px rgb(44, 7, 7),0px 6px
    ↪12px 0px" +
    ↪"rgba(44, 7, 7, 0.98),1px 23px 23px 0px rgba(44, 7, 0.85)," +
    ↪"2px" +
    ↪"51px 31px 0px rgba(44, 7, 0.5),4px 90px 36px 0px" +
    ↪"rgba(44, 7, 7, 0.15),6px 141px 40px 0px rgba(44, 7, 0.02))";
    //Фильтр
    $(".filter_window").css("background-color","rgb(244,159,47)");
    $(".container_selected_attributes").css("background-color","rgb(237,118,14)
    ↪");
    $(".select_filter").css("background-color","rgb(244,159,47)");
    //Кнопка найти
    $(".button_search_class").css("box-shadow","0px 0px 0px 0px rgb(44, 7, 7),
    ↪0px 6px 12px 0px rgba(44, 7, 7, 0.98)," +
    ↪"1px 23px 23px 0px rgba(44, 7, 7, 0.85),2px 51px 31px 0px rgba(44,
    ↪7, 7, 0.5)," +
    ↪"4px 90px 36px 0px rgba(44, 7, 7, 0.15),6px 141px 40px 0px rgba(44,
    ↪7, 7, 0.02))";
    $(".button_search_class").css("background","rgb(244,159,47)");
    $(".button_search_class").css("color","black");

    //Добавление и изменение информации о файле (Добавить, Изменить)
    $(".add_file_class").css("background-color","rgb(244,159,47)");
    $(".div_add_information_book").css("background-color","rgb(237,118,14)");
    $(".add_book_author").css("background-color","rgb(237,118,14)");
    $(".add_book_tags").css("background-color","rgb(237,118,14)");
}

```

(continues on next page)

(продолжение с предыдущей страницы)

```

$(".description_book").css("background-color","rgb(237,118,14)");

//Заметки (Открыть)
//Задний фон окна заметок
$(".container_notes").css("background-color","rgb(255,174,66)");
//Добавление и изменение (таблица заметок)
$(".table_notes").css("background-color","rgb(255,174,66)");
$(".table_notes").css("color","rgb(0,0,0)");

//Верхние кнопки
$(".menu_button").css("color","rgb(0, 0, 0)");
$(".menu_button").css("background-color","rgb(255,174,66)");

//Главное окно сайта
$(".main_window_library").css("box-shadow"," 0px 0px 0px 0px rgba(0, 0, 0,
↳12px 12px 37px 0px rgba(0, 0, 0, 0.98)," +
↳           "46px 49px 67px 0px rgba(0, 0, 0.85),104px 110px 91px
↳0px rgba(0, 0, 0, 0.5)," +
↳           "185px 195px 108px 0px rgba(0, 0, 0.15),289px 305px
↳118px 0px rgba(244,159,47, 0.02)");
$(".main_window_library").css("background","rgb(255,174,66)");

//Меню пользователя
//Задний фон меню пользователя
$(".user_menu").css("background-color","rgba(255,174,66, 1)");
//Кнопки меню пользователя с подсветкой
$(".user_icon_name").css("background-color","rgb(0, 0, 0)");
$(".user_icon_name").css("box-shadow","0px 0px 69.36000061035156px rgba(244,
↳159,47, 1)");
//Кнопки меню пользователя с подсветкой
$(".button_user_menu").css("background-color","rgb(255,174,66)");
$(".button_user_menu").css("box-shadow","0px 0px 69.36000061035156px
↳rgba(240, 255, 255, 1)");
$(".button_user_menu").css("color","black");

//Блоки книги
//Контейнеры тэгов в блоках книг
$(".tag_book").css("background-color","rgb(255,111,66)");
$(".tag_book").css("box-shadow","0px 0px 69.36000061035156px rgba(255,174,
↳66, 1)");
$(".tag_book").css("color","white");
//Контейнер описания книги
$(".text_book").css("color","black");
$(".text_book").css("background-color","rgb(227,146,79)");
$(".text_book").css("box-shadow","0px 0px 0px 0px rgb(0, 0, 0),11px 12px
↳37px 0px" +
↳           "rgba(127, 107, 107, 0.98),46px 49px 67px 0px " +
↳           "rgba(0, 0, 0, 0.85),12px 25px 22px 0px rgba(0, 0, 0, 0.5)," +

```

(continues on next page)

(продолжение с предыдущей страницы)

```

    "185px" +
→195px 108px 0px rgba(0, 0, 0, 0.15),22px 22px 22px 0px rgba(0, 0, 0, 0.02));
//Название книги
$(".label_title").css("color","rgb(255, 255, 255)");
$(".label_title").css("background-color","rgb(237,118,14)");
$(".label_title").css("box-shadow","0px 0px 69.36000061035156px rgba(244,
←159,47, 0.8)");
//Автор книги
$(".label_author").css("color","rgb(255, 255, 255)");
$(".label_author").css("background-color","rgb(244,159,47");
$(".label_author").css("background-color","0px 0px 69.36000061035156px" +
←rgba(240, 44, 5, 1));
//Тэги
$(".tag_container").css("background-color","rgb(237,118,14)");
$(".tag_container").css("box-shadow","0px 0px 69.36000061035156px rgba(227,
←146,79, 1)");
$(".tag_container").css("color","black");
//Один блок книги
$(".books_block").css("box-shadow","0px 0px 0px 0px rgb(0, 0, 0),11px 12px" +
←37px 0px rgba(0, 0, 0, 0.98)," +
→"46px 49px 67px 0px rgba(0, 0, 0, 0.85),12px 25px 22px 0px rgba(0, 0, 0,
←5)," +
→"185px 195px 108px 0px rgba(0, 0, 0, 0.15),22px 22px 22px 0px rgba(0, 0, 0,
←0.02);
$(".books_block").css("background","rgba(198,97,12, 1)");
//Контейнер информации книги
$(".container_book_info").css("background","rgba(255, 255, 255, 0.45"));

};

}

```

```

/**
 * Функция изменения задания стиля 2
 * @function style2
 * Меняет стиль некоторых элементов сайта.
*/
function style2(){
    // Класс кнопок low_button (все кнопки кроме верхних и кнопок меню
←пользователя)
    $(".low_button").css("color","rgb(0, 0, 0)");
    $(".low_button").css("background-color","rgba(128, 128, 128, 1)");
    $(".low_button").css("box-shadow","0px 0px 0px 0px rgb(44, 22, 7),0px 6px" +
←12px 0px" +
→"rgba(44, 7, 7, 0.2),1px 23px 23px 0px rgba(44, 7, 7, 0.2)," +
→"2px 51px 31px 0px rgba(44, 7, 7, 0.5),4px 90px 36px 0px" +
→"rgba(44, 7, 7, 0.15),6px 141px 40px 0px rgba(44, 7, 7, 0.02));
//Фильтр
$(".filter_window").css("background-color","rgb(157,161,170)");

```

(continues on next page)

(продолжение с предыдущей страницы)

```

        $(".container_selected_attributes").css("background-color", "rgb(157,161,170)
        ");
        $(".select_filter").css("background-color", "rgb(222,211,213)");
        //Кнопка найти
        $(".button_search_class").css("box-shadow", "0px 0px 0px 0px rgb(44, 7, 7),
        0px 6px 12px 0px rgba(44, 7, 7, 0.98), " +
        "1px 23px 23px 0px rgba(44, 7, 7, 0.85), 2px 51px 31px 0px rgba(44,
        7, 7, 0.5), " +
        "4px 90px 36px 0px rgba(44, 7, 7, 0.15), 6px 141px 40px 0px rgba(44,
        7, 7, 0.02)");
        $(".button_search_class").css("background", "rgba(157,161,170, 1)");
        $(".button_search_class").css("color", "black");

        //Добавление и изменение информации о файле (Добавить, Изменить)
        $(".add_file_class").css("background-color", "rgb(157,161,170)");
        $(".div_add_information_book").css("background-color", "rgba(55,55,55, 1)");
        $(".add_book_author").css("background-color", "rgba(55,55,55, 1)");
        $(".add_book_tags").css("background-color", "rgba(55,55,55, 1)");
        $(".description_book").css("background-color", "rgba(55,55,55, 1)");

        //Заметки (Открыть)
        //Задний фон окна заметок
        $(".container_notes").css("background-color", "rgb(157,161,170)");
        //Добавление и изменение (таблица заметок)
        $(".table_notes").css("background-color", "rgb(157,161,170)");
        $(".table_notes").css("color", "rgb(0, 0, 0)");
        //Верхние кнопки
        $(".menu_button").css("color", "rgb(0, 0, 0)");
        $(".menu_button").css("background-color", "rgb(157,161,170)");
        //Главное окно сайта
        $(".main_window_library").css("box-shadow", " 0px 0px 0px 0px rgb(0, 0, 0),
        12px 12px 37px 0px rgba(0, 0, 0, 0.98), " +
        "46px 49px 67px 0px rgba(0, 0, 0, 0.85), 104px 110px 91px
        0px rgba(0, 0, 0, 0.5), " +
        "185px 195px 108px 0px rgba(0, 0, 0, 0.15), 289px 305px
        118px 0px rgba(0, 0, 0, 0.02)");
        $(".main_window_library").css("background", "linear-gradient(to bottom,
        rgb(123, 115, 110), rgba(100, 100, 100, 0.98), " +
        "rgb(55, 55, 55), rgb(45 45 45 / 98%), rgb(65, 65, 65))");

        //Меню пользователя
        //Задний фон меню пользователя
        $(".user_menu").css("background-color", "rgba(157,161,170, 1)");
        //Изображение пользователя
        $(".user_icon_name").css("background-color", "rgba(157,161,170, 1)");
        $(".user_icon_name").css("box-shadow", "0px 0px 69.36000061035156px rgba(240,
        244, 225, 1)");

```

(continues on next page)

(продолжение с предыдущей страницы)

```
//Кнопки меню пользователя с подсветкой
$(".button_user_menu").css("background-color","rgb(157,161,170)");
$(".button_user_menu").css("box-shadow","0px 0px 69.36000061035156px
 ↪rgba(240, 255, 255, 1)");
$(".button_user_menu").css("color","black");

//Блоки книги
//Контейнеры этого в блоках книги
$(".tag_book").css("background-color","rgb(55,111,89)");
$(".tag_book").css("box-shadow","0px 0px 69.36000061035156px rgba(33,33,89,
 ↪1)");
$(".tag_book").css("color","white");
//Контейнер описания книги
$(".text_book").css("color","white");
$(".text_book").css("background-color","rgb(111, 111, 111)");
$(".text_book").css("box-shadow","0px 0px 0px rgb(0, 0, 0),11px 12px
 ↪37px 0px" +
↪"rgba(127, 107, 107, 0.98),46px 49px 67px 0px" +
↪"rgba(0, 0, 0, 0.85),12px 25px 22px 0px rgba(0, 0, 0, 0.5)," +
"185px
↪195px 108px 0px rgba(0, 0, 0, 0.15),22px 22px 22px 0px rgba(0, 0, 0, 0.02)");
//Название книги
$(".label_title").css("color","rgb(0, 0, 0)");
$(".label_title").css("background-color","rgb(157,161,170)");
$(".label_title").css("box-shadow","0px 0px 69.36000061035156px rgba(22, 22,
 ↪22, 1)");
//Автор книги
$(".label_author").css("color","rgb(255, 255, 255)");
$(".label_author").css("background-color","rgb(65, 65, 55)");
$(".label_author").css("box-shadow","0px 0px 69.36000061035156px rgba(22,
 ↪22, 22, 1)");
//Тэги
$(".tag_container").css("background-color","rgb(33,33,33)");
$(".tag_container").css("box-shadow","0px 0px 69.36000061035156px rgba(240,
 ↪255, 255, 1)");
$(".tag_container").css("color","white");
//Один блок книги
$(".books_block").css("box-shadow","0px 0px 0px rgb(0, 0, 0),11px 12px
 ↪37px 0px rgba(0, 0, 0, 0.98)," +
↪"46px 49px 67px 0px rgba(0, 0, 0, 0.85),12px 25px 22px 0px rgba(0, 0, 0,
 ↪0.5)," +
↪"185px 195px 108px 0px rgba(0, 0, 0, 0.15),22px 22px 22px 0px rgba(0, 0, 0,
 ↪0.02)");
$(".books_block").css("background","rgba(65, 65, 65, 0.45)");
$(".books_block").css("background-color","rgba(65, 65, 65, 0.45)");
//Контейнер информации книги
$(".container_book_info").css("background","rgba(188, 188, 188, 0.45));
```

(continues on next page)

(продолжение с предыдущей страницы)

};

```
/*
 * Функция изменения задания стиля 3
 * @function style3
 * Меняет стиль некоторых элементов сайта.
*/
function style3(){
    // Класс кнопок low_button (все кнопки кроме верхних и кнопок меню
    // пользователя)
    $(".low_button").css("color","rgb(255, 255, 255)");
    $(".low_button").css("background-color","rgb(49,54,89)");
    $(".low_button").css("box-shadow","0px 0px 0px 0px rgb(44, 7, 7),0px 6px
    ↵12px 0px" +
    ↵"rgba(44, 7, 7, 0.98),1px 23px 23px 0px rgba(44, 7, 0.85)," +
    ↵"2px" +
    ↵"51px 31px 0px rgba(44, 7, 0.5),4px 90px 36px 0px" +
    ↵"rgba(44, 7, 7, 0.15),6px 141px 40px 0px rgba(44, 7, 0.02))";

    //Фильтр
    $(".filter_window").css("background-color","rgb(49,54,89)");
    $(".container_selected_attributes").css("background-color","rgb(49,54,89)");
    $(".select_filter").css("background-color","rgb(222,211,213)");
    //Кнопка найти
    $(".button_search_class").css("box-shadow","0px 0px 0px 0px rgb(44, 7, 7),
    ↵0px 6px 12px 0px rgba(44, 7, 7, 0.98)," +
    ↵"1px 23px 23px 0px rgba(44, 7, 7, 0.85),2px 51px 31px 0px rgba(44,
    ↵7, 7, 0.5)," +
    ↵"4px 90px 36px 0px rgba(44, 7, 7, 0.15),6px 141px 40px 0px rgba(44,
    ↵7, 7, 0.02))";
    $(".button_search_class").css("background","rgb(49,54,89)");
    $(".button_search_class").css("color","white");

    //Добавление и изменение информации о файле (Добавить, Изменить)
    $(".add_file_class").css("background-color","rgb(49,54,89)");
    $(".div_add_information_book").css("background-color","rgb(33,22,42)");
    $(".add_book_author").css("background-color","rgb(33,22,42)");
    $(".add_book_tags").css("background-color","rgb(33,22,42)");
    $(".description_book").css("background-color","rgb(33,22,42)");

    //Заметки (Открыть)
    //Задний фон окна заметок
    $(".container_notes").css("background-color","rgb(49,54,89)");
    //Добавление и изменение (таблица заметок)
    $(".table_notes").css("background-color","rgb(49,54,89)");
    $(".table_notes").css("color","rgb(255,255,255)");

    //Верхние кнопки
}
```

(continues on next page)

(продолжение с предыдущей страницы)

```

$(".menu_button").css("color", "white");
$(".menu_button").css("background-color", "rgb(55,66,111)");

//Главное окно сайта
$(".main_window_library").css("box-shadow", " 0px 0px 0px 0px rgb(0, 0, 0),
→ 12px 12px 37px 0px rgba(0, 0, 0, 0.98), " +
→
→           "46px 49px 67px 0px rgba(49,54,89, 0.85),104px 110px 91px" +
→
→           "185px 195px 108px 0px rgba(49,54,89, 0.15),289px 305px" +
→ 118px 0px rgba(0, 0, 0, 0.02)");
$(".main_window_library").css("background", "linear-gradient(to bottom,
→ rgb(49,54,89), rgba(49,54,89, 0.98)," + " rgb(49,54,89), rgb(49,54,89))");

//Меню пользователя
//Задний фон меню пользователя
$(".user_menu").css("background-color", "rgba(55, 111, 111, 1)");
//Изображение пользователя
$(".user_icon_name").css("background-color", "rgb(49,54,89)");
$(".user_icon_name").css("box-shadow", "0px 0px 69.36000061035156px rgba(240,
→ 244, 225, 1)");
//Кнопки меню пользователя с подсветкой
$(".button_user_menu").css("background-color", "rgb(66,66,89)");
$(".button_user_menu").css("box-shadow", "0px 0px 69.36000061035156px" +
→ rgba(240, 255, 255, 1));
$(".button_user_menu").css("color", "white");

//Блоки книги
//Контейнеры тэгов в блоках книг
$(".tag_book").css("background-color", "rgb(55,111,89)");
$(".tag_book").css("box-shadow", "0px 0px 69.36000061035156px rgba(33,33,89,
→ 1)");
$(".tag_book").css("color", "white");
//Контейнер описания книги
$(".text_book").css("color", "white");
$(".text_book").css("background-color", "rgb(33,33,77)");
$(".text_book").css("box-shadow", "0px 0px 0px 0px rgb(0, 0, 0),11px 12px" +
→ 37px 0px" +
→ "rgba(127, 107, 107, 0.98),46px 49px 67px 0px" +
→ "rgba(0, 0, 0, 0.85),12px 25px 22px 0px rgba(0, 0, 0, 0.5)," +
→
→           "185px" +
→ 195px 108px 0px rgba(0, 0, 0, 0.15),22px 22px 22px 0px rgba(0, 0, 0, 0.02));
//Название книги
$(".label_title").css("color", "black");
$(".label_title").css("background-color", "rgb(211,211,222)");
$(".label_title").css("box-shadow", "0px 0px 69.36000061035156px rgba(255,
→ 216,111, 0.8)");
//Автор книги
$(".label_author").css("color", "white");

```

(continues on next page)

(продолжение с предыдущей страницы)

```

$(".label_author").css("background-color","rgb(49,54,89)");
$(".label_author").css("box-shadow","0px 0px 69.36000061035156px rgba(222, 172, 172, 0.8)");
//Тэги
$(".tag_container").css("background-color","rgb(33,33,89)");
$(".tag_container").css("box-shadow","0px 0px 69.36000061035156px rgba(240, 255, 255, 1)");
$(".tag_container").css("color","white");
//Один блок книги
$(".books_block").css("box-shadow","0px 0px 0px rgb(0, 0, 0),11px 12px 37px 0px rgba(0, 0, 0, 0.98)," +
"46px 49px 67px 0px rgba(0, 0, 0, 0.85),12px 25px 22px 0px rgba(0, 0, 0, 0.5)," +
"185px 195px 108px 0px rgba(0, 0, 0, 0.15),22px 22px 22px 0px rgba(0, 0, 0, 0.02)" );
$(".books_block").css("background","rgba(49,54,89, 0.45)");
$(".books_block").css("background-color","rgba(49,54,89, 0.45)");
//Контейнер информации книги
$(".container_book_info").css("background","rgba(49,54,89, 0.7"));

};

```

```

/**
 * Функция изменения задания стиля 4
 * @function style4
 * Меняет стиль некоторых элементов сайта.
*/
function style4(){
    // Класс кнопок low_button (все кнопки кроме верхних и кнопок меню пользователя)
    $(".low_button").css("color","rgb(255, 255, 255)");
    $(".low_button").css("background-color","rgba(61,100,45, 1)");
    $(".low_button").css("box-shadow","0px 0px 0px 0px rgb(44, 22, 7),0px 6px 12px 0px" +
"rgba(44, 7, 7, 0.2),1px 23px 23px 0px rgba(44, 7, 7, 0.2)," +
"51px 31px 0px rgba(44, 7, 7, 0.5),4px 90px 36px 0px" +
"rgba(61,100,45, 0.15),6px 141px 40px 0px rgba(61,100,45, 0.02))";

    //Фильтр
    $(".filter_window").css("background-color","rgb(112,245,41)");
    $(".container_selected_attributes").css("background-color","rgb(112,245,41"));
    $(".select_filter").css("background-color","rgb(222,211,213)");
    //Кнопка найти
    $(".button_search_class").css("box-shadow","0px 0px 0px 0px rgb(44, 7, 7),
0px 6px 12px 0px rgba(44, 7, 7, 0.98)," +

```

(continues on next page) □

(продолжение с предыдущей страницы)

```

    ↵         "1px 23px 23px 0px rgba(44, 7, 7, 0.85),2px 51px 31px 0px rgba(44, 7, 7, 0.5)," +
    ↵         "4px 90px 36px 0px rgba(44, 7, 7, 0.15),6px 141px 40px 0px rgba(44, 7, 7, 0.02)");
    $(".button_search_class").css("background","rgb(61,100,45)");
    $(".button_search_class").css("color","black");

    //Добавление и изменение информации о файле (Добавить, Изменить)
    $(".add_file_class").css("background-color","rgb(61,100,45)");
    $(".div_add_information_book").css("background-color","rgba(112,188,41, 1)");
    $(".add_book_author").css("background-color","rgba(112,188,41, 1)");
    $(".add_book_tags").css("background-color","rgba(112,188,41, 1)");
    $(".description_book").css("background-color","rgba(112,188,41, 1)");

    //Заметки (Открыть)
    //Задний фон окна заметок
    $(".container_notes").css("background-color","rgb(112,245,41)");
    //Добавление и изменение (таблица заметок)
    $(".table_notes").css("background-color","rgb(112,245,41)");
    $(".table_notes").css("color","rgb(22,22,22)");

    //Верхние кнопки
    $(".menu_button").css("color","rgb(255, 255, 255)");
    $(".menu_button").css("background-color","rgb(61,100,45)");

    //Главное окно сайта
    $(".main_window_library").css("box-shadow"," 0px 0px 0px 0px rgb(0, 0, 0),
    ↵12px 12px 37px 0px rgba(0, 0, 0, 0.98)," +
    ↵         "46px 49px 67px 0px rgba(0, 0, 0, 0.85),104px 110px 91px 0px
    ↵rgba(0, 0, 0, 0.5)," +
    ↵         "185px 195px 108px 0px rgba(0, 0, 0, 0.15),289px 305px 118px 0px
    ↵rgba(0, 0, 0, 0.02)");
    $(".main_window_library").css("background","rgb(61,100,45)");

    //Меню пользователя
    //Задний фон меню пользователя
    $(".user_menu").css("background-color","rgba(112,245,41, 1)");
    //Изображение пользователя
    $(".user_icon_name").css("background-color","rgb(61,100,45)");
    $(".user_icon_name").css("box-shadow","0px 0px 69.36000061035156px rgba(240,
    ↵ 244, 225, 1)");
    //Кнопки меню пользователя с подсветкой
    $(".button_user_menu").css("background-color","rgb(61,100,45)");
    $(".button_user_menu").css("box-shadow","0px 0px 69.36000061035156px
    ↵rgba(61,100,45, 1)");
    $(".button_user_menu").css("color","white");

    //Блоки книги

```

(continues on next page)

(продолжение с предыдущей страницы)

```

//Контейнеры тэгов в блоках книг
$(".tag_book").css("background-color", "rgb(61,100,45)");
$(".tag_book").css("box-shadow", "0px 0px 69.36000061035156px rgba(22,22,22,0.1)");
$(".tag_book").css("color", "white");
//Контейнер описания книги
$(".text_book").css("color", "white");
$(".text_book").css("background-color", "rgb(33,110,67)");
$(".text_book").css("box-shadow", "0px 0px 0px rgb(0, 0, 0),11px 12px rgba(61,100,45, 0.37px 0px" +
"rgba(61,100,45, 0.98),46px 49px 67px 0px" +
"rgba(61,100,45, 0.85),12px 25px 22px 0px rgba(0, 0, 0, 0.5)," +
"185px" +
"195px 108px 0px rgba(61,100,45, 0.15),22px 22px 22px 0px rgba(61,100,45, 0.02)");
//Название книги
$(".label_title").css("color", "rgb(0, 0, 0)");
$(".label_title").css("background-color", "rgb(112,245,41)");
$(".label_title").css("box-shadow", "0px 0px 69.36000061035156px rgba(210,111, 30)");
//Автор книги
$(".label_author").css("color", "rgb(14, 14, 14)");
$(".label_author").css("background-color", "rgb(112,177,41)");
$(".label_author").css("box-shadow", "0px 0px 69.36000061035156px rgba(177,122, 35, 1)");
//Тэги
$(".tag_container").css("background-color", "rgb(96,225,55)");
$(".tag_container").css("box-shadow", "0px 0px 69.36000061035156px rgba(22,22,22, 1)");
$(".tag_container").css("color", "black");
//Один блок книги
$(".books_block").css("box-shadow", "0px 0px 0px rgb(0, 0, 0),11px 12px" +
"37px 0px rgba(0, 0, 0, 0.98)," +
"46px 49px 67px 0px rgba(0, 0, 0, 0.85),12px 25px 22px 0px rgba(0, 0, 0, 0.5)," +
"185px 195px 108px 0px rgba(0, 0, 0, 0.15),22px 22px 22px 0px rgba(0, 0, 0, 0.02)");
$(".books_block").css("background", "rgba(61,100,45, 0.45)");
//Контейнер информации книги
$(".container_book_info").css("background", "rgba(11,111,45, 0.9");
};


```

```

/**
 * Функция изменения задания стиля 5
 * @function style5
 * Меняет стиль некоторых элементов сайта.
*/
function style5(){

```

(continues on next page)

(продолжение с предыдущей страницы)

```

// Класс кнопок low_button (все кнопки кроме верхних и кнопок меню пользователя)
$(".low_button").css("color", "rgb(255, 255, 255)");
$(".low_button").css("background-color", "rgba(236,65,14, 1)");
$(".low_button").css("box-shadow", "0px 0px 0px 0px rgb(44, 22, 7),0px 6px 12px 0px" +
"rgba(44, 7, 7, 0.2),1px 23px 23px 0px rgba(44, 7, 7, 0.2)," +
"51px 31px 0px rgba(44, 7, 7, 0.5),4px 90px 36px 0px" +
"rgba(44, 7, 7, 0.15),6px 141px 40px 0px rgba(44, 7, 7, 0.02))";

//Фильтр
$(".filter_window").css("background-color", "rgb(22, 22, 22)");
$(".container_selected_attributes").css("background-color", "rgb(22, 22, 22)" );
$(".select_filter").css("background-color", "rgb(222,211,213)");
//Кнопка найти
$(".button_search_class").css("box-shadow", "0px 0px 0px 0px rgb(236,65,14),
0px 6px 12px 0px" +
"rgba(44, 7, 7, 0.98),1px 23px 23px 0px rgba(236,65,14, 0.85)," +
"2px 51px 31px 0px rgba(44, 7, 7, 0.5),4px 90px 36px 0px rgba(44,
7, 7, 0.15),6px 141px 40px 0px rgba(44, 7, 7, 0.02))";
$(".button_search_class").css("background", "rgb(236,65,14)");
$(".button_search_class").css("color", "white");

//Добавление и изменение информации о файле (Добавить, Изменить)
$(".add_file_class").css("background-color", "rgb(22, 22, 22)");
$(".div_add_information_book").css("background-color", "rgba(22,22,22, 1)");
$(".add_book_author").css("background-color", "rgba(22,22,22, 1)");
$(".add_book_tags").css("background-color", "rgba(22,22,22, 1)");
$(".description_book").css("background-color", "rgba(22,22,22, 1)");

//Заметки (Открыть)
//Задний фон окна заметок
$(".container_notes").css("background-color", "rgb(22, 22, 22));
//Добавление и изменение (таблица заметок)
$(".table_notes").css("background-color", "rgb(22, 22, 22)");
$(".table_notes").css("color", "rgb(255,255,255)");

//Верхние кнопки
$(".menu_button").css("color", "rgb(255, 255, 255)");
$(".menu_button").css("background-color", "rgb(236,65,14)");

//Главное окно сайта
$(".main_window_library").css("box-shadow", " 0px 0px 0px 0px rgb(0, 0, 0),
12px 12px 37px 0px" +
"rgba(0, 0, 0, 0.98),46px 49px 67px 0px rgba(0, 0, 0, 0.85)," +

```

(continues on next page)

(продолжение с предыдущей страницы)

```

→           "104px 110px 91px 0px rgba(0, 0, 0, 0.5),185px 195px 108px 0px" +
→           "rgba(0, 0, 0, 0.15),289px 305px 118px 0px rgba(0, 0, 0, 0.02)");
$(".main_window_library").css("background","rgb(22, 22, 22)");

//Меню пользователя
//Задний фон меню пользователя
$(".user_menu").css("background-color","rgba(22, 22, 22, 1)");
//Изображение пользователя
$(".user_icon_name").css("background-color","rgb(0, 0, 0)");
$(".user_icon_name").css("box-shadow","0px 0px 69.36000061035156px rgba(236,
→65,14, 1)");
//Кнопки меню пользователя с подсветкой
$(".button_user_menu").css("background-color","rgb(236,65,14)");
$(".button_user_menu").css("box-shadow","0px 0px 69.36000061035156px
→rgba(22,22,22, 1)");
$(".button_user_menu").css("color","white");

//Блоки книги
//Контейнеры тэгов в блоках книг
$(".tags_book").css("background-color","rgb(22,22,22)");
$(".tags_book").css("box-shadow","0px 0px 69.36000061035156px rgba(33,33,89,
→ 1)");
$(".tags_book").css("color","white");
//Контейнер описания книги
$(".text_book").css("color","white");
$(".text_book").css("background-color","rgb(11, 11, 11)");
$(".text_book").css("box-shadow","0px 0px 0px 0px rgb(0, 0, 0),11px 12px
→37px 0px" +
→"rgba(127, 107, 107, 0.98),46px 49px 67px 0px" +
→"rgba(0, 0, 0, 0.85),12px 25px 22px 0px rgba(0, 0, 0, 0.5)," +
→"185px
→195px 108px 0px rgba(0, 0, 0, 0.15),22px 22px 22px 0px rgba(0, 0, 0, 0.02)");
//Название книги
$(".label_title").css("color","rgb(255, 255, 255)");
$(".label_title").css("background-color","rgb(22, 22, 22)");
$(".label_title").css("box-shadow","0px 0px 69.36000061035156px rgba(236,65,
→14, 0.6)");
//Автор книги
$(".label_author").css("color","rgb(255, 255, 255)");
$(".label_author").css("background-color","rgb(44, 44, 44)");
$(".label_author").css("box-shadow","0px 0px 69.36000061035156px rgba(236,
→65,14, 0.3)");
//Тэги
$(".tag_container").css("background-color","rgb(236,65,14)");
$(".tag_container").css("box-shadow","0px 0px 69.36000061035156px rgba(22,
→22, 22, 1)");
$(".tag_container").css("color","black");
//Один блок книги

```

(continues on next page)

(продолжение с предыдущей страницы)

```
$( ".books_block" ).css("box-shadow", "0px 0px 0px 0px rgb(0, 0, 0), 11px 12px ↴  
↳ 37px 0px" +  
↳ "rgba(0, 0, 0, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), " +  
↳ "12px 25px 22px 0px rgba(0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0,  
↳ 15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02)");  
$( ".books_block" ).css("background", "rgba(22,22,22, 1)");  
//Контейнер информации книги  
$( ".container_book_info" ).css("background", "rgba(11, 11, 11, 0.7"));  
};
```


CSS код

6.1 add_file.css

```
/**  
 * Кон테йнер добавления файла  
 */  
.add_file_class{  
    position: relative;  
    top: -805px;  
    left: 375px;  
    width: 1150px;  
    height: 860px;  
    background-color: rgb(22, 22, 22);  
    border-color: black;  
    border-width: 1px;  
    border-style: solid;  
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 11px 12px 37px 0px rgba(0, 0, 0, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);  
}
```

```
/**  
 * Контеинер информации о книге  
 */  
.div_add_information_book{  
    position: relative;  
    top: 15px;  
    left: 35px;  
    width: 320px;  
    background-color: rgb(22, 22, 22);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0), 11px 12px 37px 0px rgba(0, 0, 0, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);  
}
```

```
/**  
 * Класс текстовых полей добавления файла  
 */  
.input_add_book_class{  
color: rgb(0, 0, 0);  
font-size: 26px;  
width: 300px;  
}
```

```
/**  
 * Класс заголовков текстовых полей добавления файла  
 */  
.label_add_book{  
position: relative;  
font-size: 26px;  
color: rgb(255, 255, 255);  
}
```

```
/**  
 * Заголовок названия книги  
 */  
.p_add_title_book{  
top: 10px;  
left: 10px;  
}
```

```
/**  
 * Поле ввода названия книги  
 */  
.input_title_book{  
position: relative;  
top: -15px;  
left: 10px;  
}
```

```
/**  
 * Заголовок количества страниц книги  
 */  
.p_add_page_book{  
top: -20px;  
left: 10px;  
}
```

```
/**
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/* Поле ввода количества страниц книги
*/
.input_page_book{
    position: relative;
    top: -45px;
    left: 10px;
}
```

```
/***
* Заголовок путь к книге
*/
.p_add_path_book{
    top:-55px;
    left: 10px;
}
```

```
/***
* Поле ввода пути к файлу книги
*/
.input_path_book{
    position: relative;
    top: -80px;
    left: 10px;
}
```

```
/***
* Заголовок путь к изображению книги
*/
.p_add_image_book{
    top:-85px;
    left: 10px;
}
```

```
/***
* Поле ввода к изображению книги
*/
.input_image_book{
    position: relative;
    top: -110px;
    left: 10px;
}
```

```
/***
* Класс раскрывающихся списков добавления книги
*/
.add_book_combobox{
    position: relative;
    height: 40px;
    font-size: 20px;
    width: 300px;
}
```

```
/**  
 * Раскрывающий список выбора категории книги  
 */  
.combobox_category{  
    top: -75px;  
    left: 10px;  
}
```

```
/**  
 * Раскрывающий список выбора языка книги  
 */  
.combobox_language{  
    top: -45px;  
    left: 10px;  
}
```

```
/**  
 * Раскрывающий список выбора языка книги  
 */  
.combobox_year{  
    top: -15px;  
    left: 10px;  
}
```

```
/**  
 * Контейнер Авторы  
 */  
.add_book_author{  
    position: relative;  
    left: 395px;  
    top:-635px;  
    font-size: 26px;  
    width: 340px;  
    background-color: rgb(22, 22, 22);  
    height: 625px;  
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 11px 12px 37px 0px rgba(0, 0, 0, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);  
}
```

```
/**  
 * Заголовок фамилия автора  
 */  
.surname_author_label{  
    position: relative;  
    top: 5px;  
    left: 15px;  
    width: 300px;  
}
```

```
/***
 * Текстовое поле фамилии автора
 */
.surname_author_input{
    position: relative;
    top: -20px;
    left: 15px;
}
```

```
/***
 * Заголовок имени автора
 */
.name_author_label{
    position: relative;
    top:-25px;
    left: 15px;
}
```

```
/***
 * Текстовое поле имени автора
 */
.name_author_input{
    position: relative;
    top: -50px;
    left: 15px;
}
```

```
/***
 * Заголовок отчества автора
 */
.patronymic_author_label{
    position: relative;
    top:-55px;
    left: 15px;
}
```

```
/***
 * Текстовое поле отчества автора
 */
.patronymic_author_input{
    position: relative;
    top: -80px;
    left: 15px;
}
```

```
/***
 * Кнопка Добавить нового автора
 */
.button_add_new_author{
    width: 300px;
    height: 40px;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    font-size: 21px;  
    left: 15px;  
    top: 355px;  
}
```

```
/**  
 * Заголовок Список авторов для книги  
 */  
.title_list_authors{  
    left: 15px;  
    top: -35px;  
}
```

```
/**  
 * Класс мульти списка добавления книги  
 */  
.field_combobox_add_book{  
    position: relative;  
    width: 300px;  
    height: 60px;  
    overflow: auto;  
    font-size: 18px;  
}
```

```
/**  
 * Список авторов для книги  
 */  
.field_authors{  
    left: 15px;  
    top:-55px;  
}
```

```
/**  
 * Раскрывающий список авторов  
 */  
.combobox_authors{  
    top:-35px;  
    left: 15px;  
}
```

```
/**  
 * Кнопка Добавить автора в список  
 */  
.button_add_list_author{  
    width: 300px;  
    height: 40px;  
    font-size: 21px;  
    left: 15px;  
    top: 575px;  
}
```

```
/*
 * Тэги книги
 */
.add_book_tags{
    position: relative;
    top:-1285px;
    left:780px;
    background-attachment: fixed;
    font-size: 26px;
    width: 340px;
    background-color: rgb(22, 22, 22);
    height: 455px;
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 11px 12px 37px 0px rgba(0, 0, 0, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);
}
```

```
/*
 * Заголовок Тэги книги, Добавить новый тэг, Название тэга
 */
.tag_label{
    position: relative;
    top: 5px;
    left: 15px;
    width: 300px;
}
```

```
/*
 * Поле ввода нового тэга
 */
.name_tag_input{
    position: relative;
    top: -20px;
    left: 15px;
}
```

```
/*
 * Кнопка добавления нового тэга
 */
.button_new_tag{
    width: 300px;
    height: 40px;
    font-size: 21px;
    left: 15px;
    top: 175px;
}
```

```
/*
 * Заголовок Список тэгов для книги
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/*
.label_list_tags{
    position: relative;
    top: 25px;
    left: 15px;
    width: 300px;
}
```

```
/**
* Список тэгов для книги
*/
.field_list_tags{
    left: 15px;
    top: 5px;
}
```

```
/**
* Кнопка Добавить тэг в список
*/
.button_add_list_tag{
    width: 300px;
    height: 40px;
    font-size: 21px;
    left: 15px;
    top: 395px;
}
```

```
/**
* Раскрывающий список выбора тэга
*/
.comboobox_tags{
    top: 20px;
    left: 15px;
}
```

```
/**
* Описание книги
*/
.description_book{
    position: relative;
    top: -1285px;
    left: 780px;
    background-attachment: fixed;
    font-size: 26px;
    width: 340px;
    background-color: rgb(22, 22, 22);
    height: 145px;
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 11px 12px 37px 0px rgba(0, 0, 0, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);
```

(continues on next page)

(продолжение с предыдущей страницы)

}

```
/**  
 * Заголовок Описание книги  
 */  
.description_book_label{  
    left:15px;  
}
```

```
/**  
 * Текстовое поле Описание книги  
 */  
.description_book_input{  
    white-space: pre-wrap;  
    word-wrap: normal;  
    position: relative;  
    width: 300px;  
    height: 100px;  
    left:15px;  
    top:-20px;  
    z-index: 44;  
}
```

```
/**  
 * Кнопка Добавить книгу  
 */  
.button_create_book{  
    position: relative;  
    top:-1250px;  
    left:870px;  
    width: 250px;  
}
```

6.2 book_container.css

```
/**  
 * Контейнер со всеми книгами  
 */  
.all_block_books{  
    position: absolute;  
    top:0px;  
    left:25px;  
}
```

```
/**  
 * Блок с книгой  
 */
```

(continues on next page)

(продолжение с предыдущей страницы)

```
.books_block_inside{
    position: absolute;
    width: 1154px;
    height: 422px;
    left: 344px;
    right: 25px;
    top: 322px;
}
```

```
/**
* Блок книги 1 уровня
*/
.books_block{
    width: 1154px;
    height: 420px;
    bottom: 336px;
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0), 11px 12px 37px 0px rgba(23, 22, 22, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);
    background: rgba(22, 22, 22, 0.45);
}
```

```
/**
* Блок описания книги
*/
.container_book_info{
    position: fixed;
    width: 570px;
    height: 310px;
    background: rgba(11, 11, 11, 0.45);
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0), 11px 12px 37px 0px rgba(23, 22, 22, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);
}
```

```
/**
* Кнопка Открыть
*/
.button_container_books1{
    position: relative;
    left: 590px;
    top: -58px;
    width: 200px;
    height: 60px;
    font-size: 26px;
}
```

```
/**
* Кнопка Изменить
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/*
.button_container_books2{
    position: relative;
    left: 590px;
    top: -50px;
    width: 200px;
    height: 60px;
    font-size: 26px;
}
```

```
/**
* Кнопка В избранное
*/
.button_container_books3{
    position: relative;
    left: 590px;
    top: -42px;
    width: 200px;
    height: 60px;
    font-size: 26px;
}
```

```
/**
* Кнопка На чтение
*/
.button_container_books4{
    position: relative;
    left: 590px;
    top: -34px;
    width: 200px;
    height: 60px;
    font-size: 26px;
}
```

```
/**
* Кнопка В прочитанное
*/
.button_container_books5{
    position: relative;
    left: 590px;
    top: -26px;
    width: 200px;
    height: 60px;
    font-size: 26px;
}
```

```
/**
* Описание книги
*/
.text_book{
    white-space: pre-wrap;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
word-wrap: normal;
overflow: auto;
width: 570px;
height: 220px;
font-size: 24px;
color: white;
background-color: rgb(11, 11, 11);
box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 11px 12px 37px 0px rgba(127, 107, 107, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px, 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px, 0px 0px 0px 0.02;
}
```

```
/**
* Заголовок названия книги
*/
.label_title{
    white-space: pre nowrap;
    overflow-x: auto;
    word-wrap: normal;
    position: relative;
    color: rgb(255, 255, 255);
    font-family: Inter;
    font-size: 20px;
    letter-spacing: 0px;
    text-align: center;
    font-variant: small-caps;
    width: 570px;
    height: 43px;
    background-color: rgb(22, 22, 22);
    box-shadow: 0px 0px 69.36000061035156px rgba(236, 65, 14, 0.6);
    line-height: 32px;
}
```

```
/**
* Заголовок имени автора
*/
.label_author{
    white-space: nowrap;
    overflow-x: auto;
    word-wrap: normal;
    position: relative;
    color: rgb(255, 255, 255);
    font-family: Inter;
    font-size: 18px;
    letter-spacing: 0px;
    text-align: center;
    font-variant: small-caps;
    width: 375px;
    height: 50px;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
background-color: rgb(44, 44, 44);
box-shadow: 0px 0px 69.36000061035156px rgba(236,65,14, 0.3);
line-height: 32px;
}
```

```
/*
* Заголовок языка книги и года издания
*/
.label_year_language{
    font-weight:bold ;
    position: absolute;
    left: 420px;
    top: 45px;
    color: #ffffcfc;
}
```

```
/*
* Заголовок категории книги
*/
.label_category{
    font-weight:bold ;
    position: absolute;
    left: 400px;
    top: 312px;
    color: #ffffcfc;
}
```

```
/*
* Заголовок статуса книги
*/
.label_state{
    font-weight:bold ;
    position: absolute;

    top: 312px;
    color: #ffffcfc;
}
```

```
/*
* Конейнер блоков тэгов книги
*/
.tags_book{
    white-space: pre-wrap;
    word-wrap: normal;
    overflow: auto;
    width: 570px;
    height:90px;
    font-size: 24px;
    color: black;
    scrollbar-width: thin;
    scrollbar-color: hsl(0 0% 50%);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
background-color: 0px 0px 69.36000061035156px rgba(33,33,89, 1);  
box-shadow: r;  
}
```

```
/**  
* Контейнер тэгов  
*/  
.tag_container{  
    width: auto;  
    height: 44px;  
    font-family: Inter;  
    font-size: 22px;  
    line-height: 41px;  
    white-space: nowrap;  
    color: rgb(255, 255, 255);  
    text-align: center;  
    background-color: rgb(236, 65, 14);  
    border-color: black;  
    border-width: 1px;  
    border-style: solid;  
    box-shadow: 0px 0px 69.36000061035156px rgba(22, 22, 22, 1);  
}  
}
```

```
/**  
* Дизайн скролла  
*/  
::-webkit-scrollbar {  
    height: 12px;  
    width: 12px;  
    background: #3b280d53;  
}  
  
::-webkit-scrollbar-thumb {  
    background: #c2bab5;  
    -webkit-border-radius: 1ex;  
    -webkit-box-shadow: 0px 1px 2px rgba(22, 20, 20, 0);  
}  
  
::-webkit-scrollbar-corner {  
    background: #000;  
}
```

6.3 change_file.css

```
/***
 * Класс кнопки изменения информации о книге
 */
.change_btn{
    font-size: 16px;
    width: 90px;
    height: 20px;
    line-height: 14px;
}
```

```
/***
 * Контейнер информации о книге
 */
.change_information{
    height: 690px;
}
```

```
/***
 * Кнопка изменить название книги
 */
#change_name_book{
    position: absolute;
    top:145px;
    left: 220px;
}
```

```
/***
 * Кнопка изменить количество страниц, книги
 */
#change_page_book{
    position: absolute;
    top:232px;
    left: 220px;
}
```

```
/***
 * Кнопка изменить путь к файлу книги
 */
#change_path_book{
    position: absolute;
    top:315px;
    left: 220px;
}
```

```
/***
 * Кнопка изменить путь к изображению книги
 */
#change_path_image_book{
    position: absolute;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    top:405px;  
    left: 220px;  
}
```

```
/**  
 * Кнопка изменить категорию книги  
 */  
#change_category_book{  
    position: absolute;  
    top:495px;  
    left: 220px;  
}
```

```
/**  
 * Кнопка изменить язык книги  
 */  
#change_language_book{  
    position: absolute;  
    top:580px;  
    left: 220px;  
}
```

```
/**  
 * Кнопка изменить год книги  
 */  
#change_year_book{  
    position: absolute;  
    top:665px;  
    left: 220px;  
}
```

```
/**  
 * Контейнер Авторы  
 */  
.change_author{  
    top:-700px;  
    height: 690px;  
}
```

```
/**  
 * Раскрывающийся список авторов  
 */  
.combobox_change_list_authors{  
    top:-30px;  
}
```

```
/**  
 * Кнопка Добавить автора в список  
 */  
.btn_change_list_authors{
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        top: 615px;  
    }
```

```
/**  
 * Кнопка Добавить авторов из списка  
 */  
.btn_add_list_authors{  
    position: absolute;  
    top: 515px;  
    left: 225px;  
}
```

```
/**  
 * Тэги книги  
 */  
.change_tags{  
    top:-1415px;  
    height: 485px;  
}
```

```
/**  
 * Кнопка Добавить тэг в список  
 */  
.change_btn_add_list_tag{  
    top: 425px;  
}
```

```
/**  
 * Раскрывающий список выбора тэга  
 */  
.change_combobox_tags{  
    top: 50px;  
}
```

```
/**  
 * Кнопка добавления списка тэгов  
 */  
.change_list_tags{  
    position: absolute;  
    top:340px;  
    left:225px;  
}
```

```
/**  
 * Описание книги  
 */  
.change_description{  
    top:-1420px;  
    height: 185px;  
}
```

```
/**  
 * Кнопка изменения описания книги  
 */  
#btn_change_description{  
    position: absolute;  
    left:220px;  
    top:150px;  
}
```

6.4 filter_window.css

```
/**  
 * Окно фильтра поиска  
 */  
.filter_window{  
    visibility: hidden;  
    background-color: rgba(22, 22, 22);  
    width: 1150px;  
    height: 70px;  
    z-index: 15;  
    position: relative;  
    left: 370px;  
    top: -630px;  
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 11px 12px 37px 0px rgba(23, 22, 22, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);  
}
```

```
/**  
 * Класс раскрывающегося списка фильтра поиска  
 */  
.select_filter{  
    width: 380px;  
    font-size: 21px;  
    background-color: rgb(222,211,213);  
}
```

```
/**  
 * Класс кнопок в окне Фильтр  
 */  
.button_filter{  
    height:40px;  
    width: 120px;  
    font-size: 21px;  
}
```

```
/**  
 * Кнопка применения фильтра
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/*
#enter_window_filter{
    position: absolute;
    left:1010px;
    top:30px;
}
```

6.5 main_book.css

```
/**
* класс кнопок верхнего меню
*/
.menu_button{
    position: absolute;
    top: 30px;
    width: 214px;
    height: 72px;
    font-family: Inter;
    font-size: 28px;
    font-weight: 800;
    line-height: 34px;
    letter-spacing: 0px;
    color: rgb(255, 255, 255);
    text-align: center;
    font-variant: small-caps;
    cursor: pointer;
    background-color: rgb(236, 65, 14);
    border-color: black;
    border-width: 1px;
    border-style: solid;
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 11px 12px 37px 0px rgba(0, 0, 0, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);
    opacity: 0.6;
    transition: 0.3s;
    border-radius: 10%;
}
.menu_button:hover {opacity: 1}
```

```
/**
* расположение кнопки 1
*/
#menu_button1{
    left: 25px;
}
```

```
/**
* расположение кнопки 2
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/*
#menu_button2{
    left: 345px;
}
```

```
/**
* расположение кнопки 3
*/
#menu_button3{
    left: 672px;
}
```

```
/**
* расположение кнопки 4
*/
#menu_button4{
    left: 990px;
}
```

```
/**
* расположение кнопки 5
*/
#menu_button5{
    left: 1310px;
}
```

```
/**
* Класс для всех кнопок сайта
*/
.low_button{
    color: rgb(255, 255, 255);
    font-family: Inter;
    font-size: 28px;
    font-weight: 800;
    line-height: 34px;
    letter-spacing: 0px;
    text-align: center;
    font-variant: small-caps;
    width: 158px; height: 67px;
    background-color: rgba(236,65,14, 1);
    border-radius: 9999px;
    border-color: black;
    border-width: 1px;
    border-style: solid;
    position: absolute;
    box-sizing: border-box;
    border: 1px solid rgb(0, 0, 0);
    box-shadow: 0px 0px 0px 0px rgba(44, 22, 7),0px 6px 12px 0px rgba(44, 7,
    ↵ 7, 0.2),1px 23px 23px 0px rgba(44, 7, 7, 0.2), 2px 51px 31px 0px rgba(44, 7,
    ↵ 7, 0.5),4px 90px 36px 0px rgba(44, 7, 7, 0.15),6px 141px 40px 0px rgba(44, 7,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    ↵7, 7, 0.02);
    border-radius: 10%;
    opacity: 0.6;
    transition: 0.3s;
    cursor: pointer;
}
.low_button:hover {opacity: 1}
```

```
/**
 * кнопка скрыть/показать
 * расположение кнопки показать верхнее меню
*/
.button_hide_class{
    left: 25px;
    top: 150px;
    width: 130px;
    height: 67px;
}
```

```
/**
 * расположение кнопки скрыть верхнее меню
*/
.button_show_class{
    visibility: hidden;
    left: 25px;
    top: 150px;
    width: 130px;
    height: 67px;
}
```

```
/**
 * кнопки изменения стиля сайта
*/
.button_style{
    visibility: hidden;
    position: fixed;
    top: 150px;
    left: 175px;
    opacity: 0.6;
    transition: 0.3s;
    cursor: pointer;
}
.button_style:hover {opacity: 1}
```

```
/**
 * главная кнопка изменения стиля сайта
*/
.button_style_visibility_main{
    height: 67px;
    width: 95px;
    visibility: visible;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        border-radius: 10%;  
    }
```

```
/**  
 * расположение кнопки стиля 1  
 */  
.button_style_position1{  
    background-color: rgb(214, 129, 24);  
    height: 40px;  
    width: 35px;  
    left: 30px;  
    top: 250px;  
    border-radius: 100%;  
}
```

```
/**  
 * расположение кнопки стиля 2  
 */  
.button_style_position2{  
    background-color: rgba(158, 181, 181, 0.848);  
    height: 40px;  
    width: 35px;  
    left: 80px;  
    top: 250px;  
    border-radius: 100%;  
}
```

```
/**  
 * расположение кнопки стиля 3  
 */  
.button_style_position3{  
    background-color: rgb(49,54,89);  
    height: 40px;  
    width: 35px;  
    left: 130px;  
    top: 250px;  
    border-radius: 100%;  
}
```

```
/**  
 * расположение кнопки стиля 4  
 */  
.button_style_position4{  
    background-color: rgb(112,245,41);  
    height: 40px;  
    width: 35px;  
    left: 180px;  
    top: 250px;  
    border-radius: 100%;  
}
```

```
/***
 * расположение кнопки стиля 5
 */
.button_style_position5{
    background-color: rgb(18, 20, 20);
    height: 40px;
    width: 35px;
    left: 230px;
    top: 250px;
    border-radius: 100%;
}
```

```
/***
 * главный контейнер сайта
 */
.main_window_library{
    position: relative;
    width: 1555px;
    max-height: 2500px;
    height: 2500px;
    background-attachment: fixed;
    backdrop-filter: blur(22px);
    background: rgb(22, 22, 22);
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0), 11px 12px 37px 0px rgba(23, 22, 22, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);
}
```

```
/***
 * класс кнопки фильтр
 */
.button_filter_class{
    width: 158px;
    height: 67px;
    left: 370px;
    top: 150px;
}
```

```
/***
 * поле ввода для поиска
 */
.search_input_class{
    position: absolute;
    width: 971px;
    height: 70px;
    left: 550px;
    top: 150px;
    font-size: 25px;
    box-sizing: border-box;
    border: 1px solid rgb(0, 0, 0);
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 11px 12px 37px 0px rgba(23, 22, 22, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
↳22, 22, 0.98),46px 49px 67px 0px rgba(0, 0, 0, 0.85),12px 25px 22px 0px ↳
↳rgba(0, 0, 0, 0.5),185px 195px 108px 0px rgba(0, 0, 0, 0.15),22px 22px 22px ↳
↳0px rgba(0, 0, 0, 0.02);
    background: rgb(196, 196, 196);
}
```

```
/***
 * кнопка найти
 */
.button_search_class{
    width: 158px;
    height: 67px;
    left: 1360px;
    top: 150px;
    background: rgb(236,65,14);
    color: white;
}
```

```
/***
 * Контейнер для страниц
 */
.page{
    position: relative;

    top:20px;
    left: 395px;
}
```

```
/***
 * Блок страницы
 */
.page_container{
    width: 125px;
    position: relative;
}
```

6.6 open_book.css

```
/***
 * Контейнер книги
 */
.book_window_class{
    position: relative;
    top: -585px;
    left: 375px;
    width: 1150px;
}
```

```
/***
 * Документ книги
 */
.frame_book_class{
    width: 1150px;
    height: 720px;
}
```

```
/***
 * Контейнер меню открытой книги
 */
.menu_book_read{
    position: relative;
    top: -770px;
    left: 0px
}
```

```
/***
 * Класс меню кнопок открытой книги
 */
.book_button_open{
    position:static;
    width: 180px;
    height: 44px;
    font-family: Inter;
    font-size: 22px;
    line-height: 34px;
}
```

```
/***
 * Контейнер закладки
 */
.container_marker{
    position: relative;
    visibility: hidden;
}
```

```
/***
 * Поле ввода страницы закладки
 */
.input_marker{
    width: 80px;
    height: 44px;
    font-size: 22px;
}
```

```
/***
 * Кнопка изменения страницы закладки
 */
.btn_marker{
    left: 80px;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
width: 180px;
height: 44px;
font-family: Inter;
font-size: 22px;
line-height: 34px;
}
```

```
/**
 * Контейнер заметок
 */
.container_notes{
    position: absolute;
    width: 1150px;
    height: 600px;
    top: 50px;
    background-color: rgb(22, 22, 22);
    visibility: hidden;
}
```

```
/**
 * Контейнер таблицы
 */
.container_table{
    overflow-y: scroll;
    position: absolute;
    width: 1100px;
    height: 300px;
    left: 25px;
}
```

```
/**
 * Таблица заметок
 */
.table_notes{
    width: 1080px;
    height: auto;
    background-color: rgb(22, 22, 22);
    color: rgb(255, 255, 255);
    font-size: 26px;
    word-break: break-all;
    position: relative;
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 11px 12px 37px 0px rgba(23, 22, 22, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);
}
```

```
/**
 * Класс линий границ таблицы
 */
.border_tables{
```

(continues on next page)

(продолжение с предыдущей страницы)

```

        border: 3px;
        border-color: black;
        border-style: groove;
    }
}

```

```

/**
 * Класс ячейки id
 */
.id_note{
    width: 50px;
    height: 20px;
}

```

```

/**
 * Класс ячейки заметки
 */
.name_note{
    height: 20px;
    white-space: pre-wrap;
}

```

```

/**
 * Класс записи таблицы заметок
 */
.row_table{
    height: 20px;
}

```

```

/**
 * Контеинер добавления заметок
 */
.add_notes{
    position: absolute;
    width: 1090px;
    height: 210px;
    top: 320px;
    left: 25px;
    background-color: rgb(213, 213, 229);
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 11px 12px 37px 0px rgba(23, 22, 22, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);
}

```

```

/**
 * Кнопка Добавить заметку
 */
.btn_add_note{
    width: 200px;
    height: 44px;
    font-size: 22px;
}

```

(continues on next page)

(продолжение с предыдущей страницы)

```
position: absolute;
top: 225px;
left: 900px;
}
```

```
/*
* Поле ввода заметки
*/
.field_note{
    position: relative;
    top: -20px;
    width: 1100px;
    height: 220px;
    font-size: 26px;
}
```

6.7 user_menu.css

```
/*
* меню пользователя
*/
.user_menu
{
    background-color: rgba(22, 22, 22, 1);
    position: relative;
    width: 317px;
    height: 863px;
    left: 25px;
    right: 1222px;
    top: 320px;
    bottom: -13px;
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0), 11px 12px 37px 0px rgba(23, 22, 0.98), 46px 49px 67px 0px rgba(0, 0, 0.85), 12px 25px 22px 0px rgba(0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0.15), 22px 22px 22px 0px rgba(0, 0, 0, 0.02);
    opacity: 1;
}
```

```
/*
* икона пользователя в меню пользователя
*/
.user_icon_name
{
    background-color: rgb(0, 0, 0);
    box-shadow: 0px 0px 69.36000061035156px rgba(236, 65, 14, 1);
    position: absolute;
    width: 317px;
    height: 93px;
    left: 0;
```

(continues on next page)

(продолжение с предыдущей страницы)

```

    right: 1206px;
}

/***
 * кнопка меню пользователя
*/
.user_menu_button
{
    position: absolute;
    left: 40px;
    width: 296px;
    height: 93px;
    font-size: 26px;
    font-weight: 800;
    line-height: 100%;
    text-align: center;
    color: white;
}

```

```

/***
 * кнопки меню пользователя
*/
.button_user_menu{
    position: absolute;
    background-color: rgb(236,65,14);
    box-shadow: 0px 0px 69.36000061035156px rgba(22,22,22, 1);
    width: 317px;
    height: 90px;
    color: rgb(255, 255, 255);
    font-family: Inter;
    font-size: 20px;
    letter-spacing: 0px;
    text-align: center;
    font-variant: small-caps;
    opacity: 0.6;
    transition: 0.3s;
}
.button_user_menu:hover {opacity: 1}

```

```

/***
 * расположение кнопки 1
*/
.button_user_menu_position1
{
    top:150px;
}

```

```

/***
 * расположение кнопки 2
*/

```

(continues on next page)

(продолжение с предыдущей страницы)

```
.button_user_menu_position2
{
    top:240px;
}
```

```
/**
 * расположение кнопки 3
 */
.button_user_menu_position3
{
    top:330px;
}
```

```
/**
 * расположение кнопки 4
 */
.button_user_menu_position4
{
    top:420px;
}
```

```
/**
 * расположение кнопки 5
 */
.button_user_menu_position5
{
    top:510px;
}
```

```
/**
 * изображение пользователя
 */
.image_user{
    position: absolute;
    left: 0px;
    top: 0px;
    width: 90px;
    height:95px;
    box-shadow: 0px 0px 0px 0px rgb(0, 0, 0),11px 12px 37px 0px rgba(23, 11, 22, 0.98),46px 49px 67px 0px rgba(0, 0, 0, 0.85),12px 25px 22px 0px rgba(0, 0, 0, 0.5),185px 195px 108px 0px rgba(0, 0, 0, 0.15),22px 22px 22px 0px rgba(0, 0, 0, 0.02);
}
```

6.8 index_book.css

```
/***
 * Главный контейнер страницы авторизации
 */
.window{
    position: relative;
    width: 1000px;
    height: 1200px;
    background-attachment: fixed;
    box-shadow: 0px 0px 0px 0px rgba(0, 0, 0, 0), 12px 12px 37px 0px rgba(0, 0, 0, 0.98), 46px 49px 67px 0px rgba(0, 0, 0, 0.85), 104px 110px 91px 0px rgba(0, 0, 0, 0.5), 185px 195px 108px 0px rgba(0, 0, 0, 0.15), 119px 305px 118px 0px rgba(0, 0, 0, 0.02);
    backdrop-filter: blur(22px);
    background: linear-gradient(to bottom, rgb(13, 13, 13), rgba(17, 17, 17, 0.98), rgb(8, 7, 7), rgba(24, 22, 22, 0.98), rgb(17, 15, 14));
}
```

```
/***
 * Класс контейнер для полей ввода логина и пароля
 */
.input_lk{
    width: 355px;
    height: 70px;
    box-sizing: border-box;
    border: 1px solid rgb(0, 0, 0);
    box-shadow: 0px 0px 0px 0px rgba(44, 7, 7, 0.98), 0px 6px 12px 0px rgba(44, 7, 7, 0.85), 2px 51px 31px 0px rgba(44, 7, 7, 0.5), 4px 90px 36px 0px rgba(44, 7, 7, 0.15), 6px 141px 40px 0px rgba(44, 7, 7, 0.02);
    background: rgb(196, 196, 196);
}
```

```
/***
 * Расположение контейнера ввода логина
 */
#login_input{
    position: absolute;
    left: 1123px;
    top: 217px;
}
```

```
/***
 * Расположение контейнера ввода пароля
 */
#password_input{
    position: absolute;
    left: 1123px;
    top: 417px;
}
```

```
/**  
 * Класс надписей для окна авторизации  
 */  
.label_auth{  
color:white;  
font-size: 36px;  
}
```

```
/**  
 * Надпись логин  
 */  
#label_login{  
position: absolute;  
left: 1123px;  
top: 127px;  
}
```

```
/**  
 * Надпись Пароль  
 */  
#label_password{  
position: absolute;  
left: 1123px;  
top: 327px;  
}
```

```
/**  
 * Кнопка войти  
 */  
#button_in{  
position: absolute;  
left: 1123px;  
top: 527px;  
font-size: 36px;  
width: 370px;  
text-align: center;  
color:rgb(0, 0, 0);  
font-family: Arial, Helvetica, sans-serif;  
background-color: rgb(214, 21, 21);  
cursor: pointer;  
border: 1px solid rgb(248, 246, 246);  
box-shadow: 0px 0px 0px 0px rgba(44, 7, 7, 0), 6px 12px 0px rgba(44, 7, 7, 0.98), 1px 23px 23px 0px rgba(44, 7, 7, 0.85), 2px 51px 31px 0px rgba(44, 7, 7, 0.5), 4px 90px 36px 0px rgba(44, 7, 7, 0.15), 6px 141px 40px 0px rgba(44, 7, 7, 0.02);  
}
```

```
/**  
 * Класс полей для ввода  
 */  
.text_input{
```

(continues on next page)

(продолжение с предыдущей страницы)

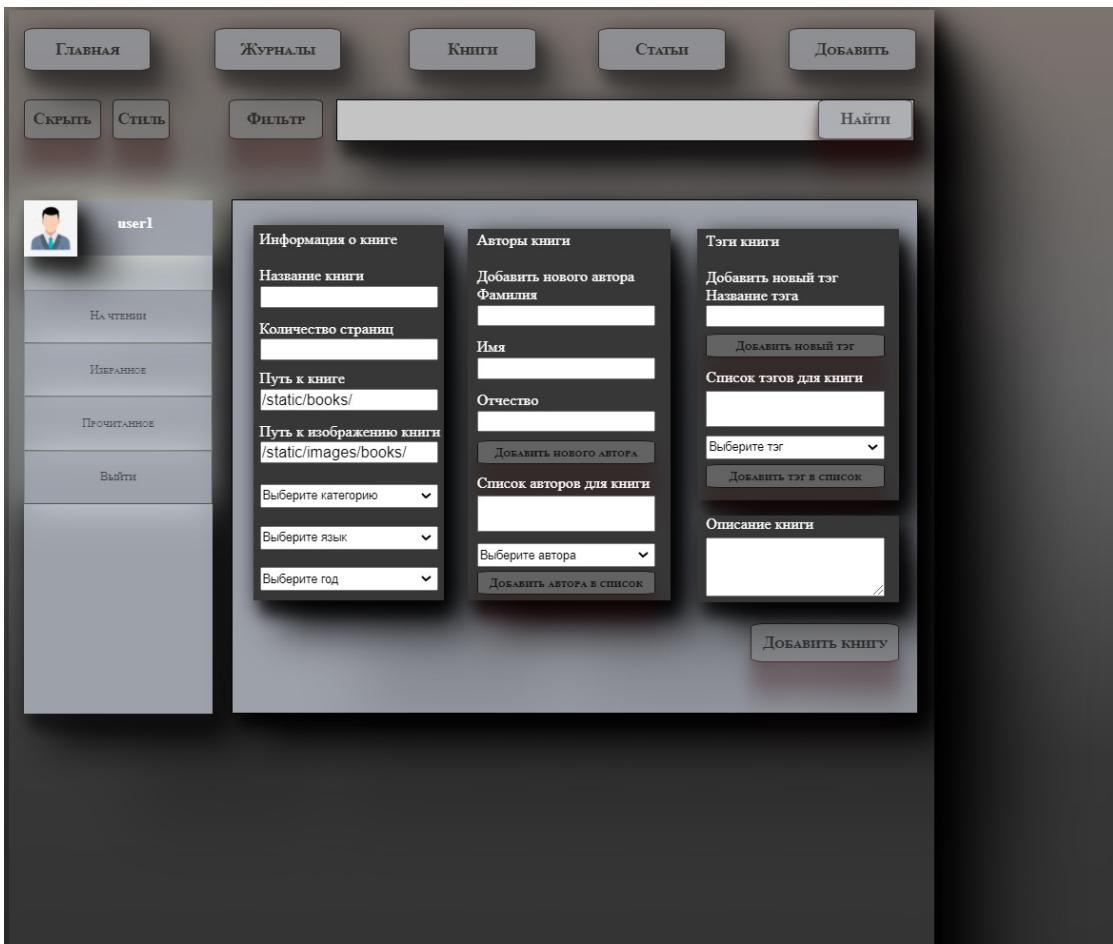
```
position: absolute;
width: 371px;
height: 70px;
font-size: 25px;
box-sizing: border-box;
border: 1px solid rgb(0, 0, 0);
box-shadow: 0px 0px 0px 0px rgb(44, 7, 7), 0px 6px 12px 0px rgba(44, 7, 7, 0.98), 1px 23px 23px 0px rgba(44, 7, 7, 0.85), 2px 51px 31px 0px rgba(44, 7, 7, 0.5), 4px 90px 36px 0px rgba(44, 7, 7, 0.15), 6px 141px 40px 0px rgba(44, 7, 7, 0.02);
background: rgb(196, 196, 196);
```


Стили приложения

Страница добавления книги (стиль 1)

The screenshot shows a web application interface for adding a book. The top navigation bar includes links for Главная, Журналы, Книги, Статьи, and Добавить. Below the navigation are buttons for Скрыть, Стиль, Фильтр, and Найти. On the left, a sidebar for user 'user1' displays sections: На чтении, Избранное, Прочитанное, and Выйти. The main content area is divided into three columns: 'Информация о книге' (Book Information), 'Авторы книги' (Authors), and 'Тэги книги' (Tags). The 'Информация о книге' column contains fields for Название книги (Book name), Количество страниц (Number of pages), Путь к книге (Path to book), Путь к изображению книги (Path to book image), Выберите категорию (Select category), Выберите язык (Select language), and Выберите год (Select year). The 'Авторы книги' column includes fields for Добавить нового автора (Add new author), Фамилия (Surname), Имя (Name), Отчество (Middle name), and a 'Список авторов для книги' (List of authors for book) section. The 'Тэги книги' column features fields for Добавить новый тэг (Add new tag), Название тэга (Tag name), a 'Список тэгов для книги' (List of tags for book) section, and a 'Вы特色е тэг' (Select tag) dropdown. A large 'Добавить книгу' (Add book) button is located at the bottom right.

Страница добавления книги (стиль 2)



Страница добавления книги (стиль 3)

The screenshot shows the 'Добавить книгу' (Add Book) page. At the top, there is a navigation bar with buttons for Главная (Home), Журналы (Journals), Книги (Books), Статьи (Articles), Добавить (Add), Скрыть (Hide), Стиль (Style), Фильтр (Filter), a search input field, and a Найти (Find) button.

On the left, a sidebar for the user 'user1' displays sections: На чтении (Reading), Избранное (Favorites), Прочитанное (Read), and Выйти (Logout).

The main form area is divided into several sections:

- Информация о книге**:
 - Название книги: input field
 - Количество страниц: input field
 - Путь к книге: input field (value: /static/books/)
 - Путь к изображению книги: input field (value: /static/images/books/)
 - Выберите категорию: dropdown menu
 - Выберите язык: dropdown menu
 - Выберите год: dropdown menu
- Авторы книги**:
 - Добавить нового автора: button
 - Фамилия: input field
 - Имя: input field
 - Отчество: input field
 - Добавить нового автора: button
 - Список авторов для книги: input field
 - Выберите автора: dropdown menu
 - Добавить автора в список: button
- Тэги книги**:
 - Добавить новый тэг: button
 - Название тэга: input field
 - Добавить новый тэг: button
 - Список тэгов для книги: input field
 - Выберите тэг: dropdown menu
 - Добавить тэг в список: button
- Описание книги**:
 - input field

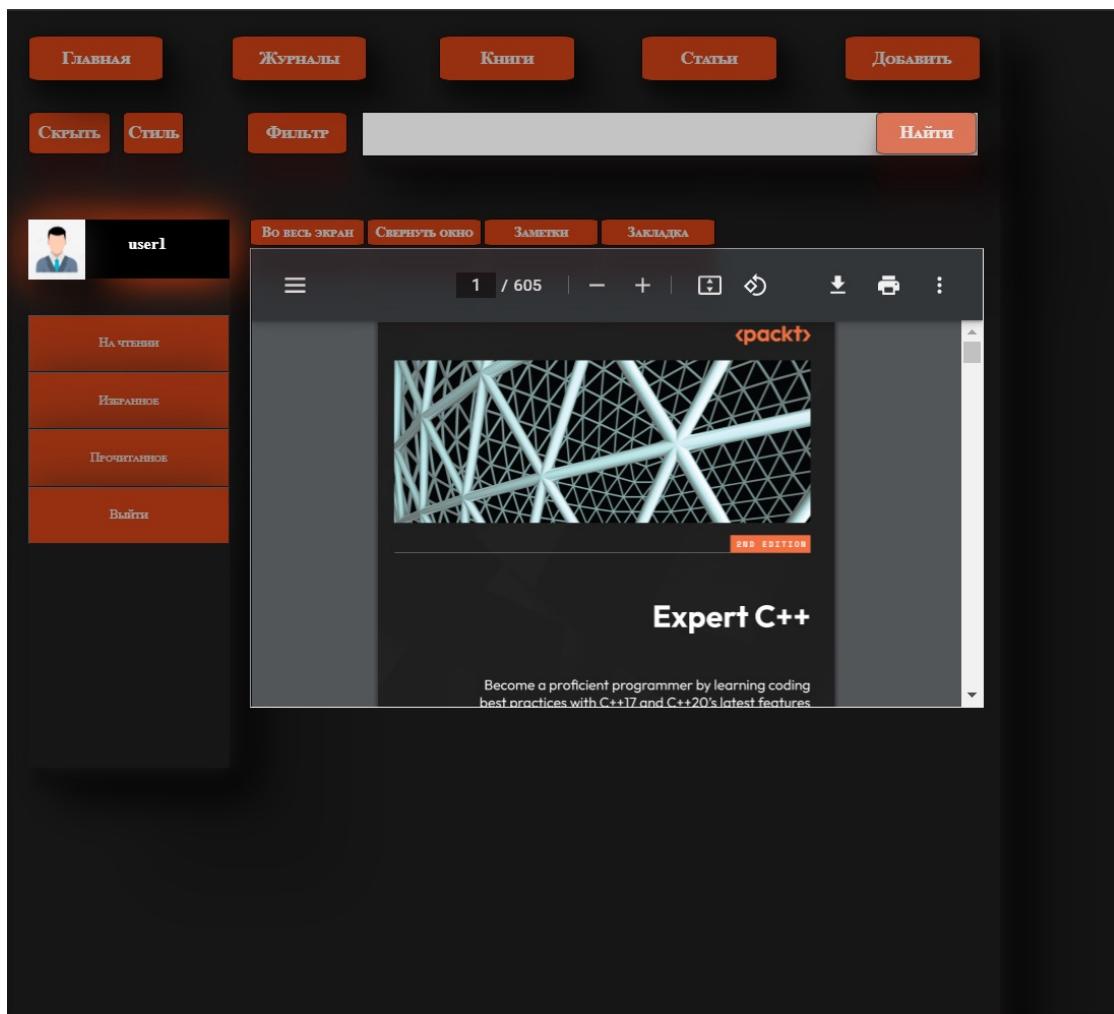
Добавить книгу: button at the bottom right.

Страница добавления книги (стиль 4)

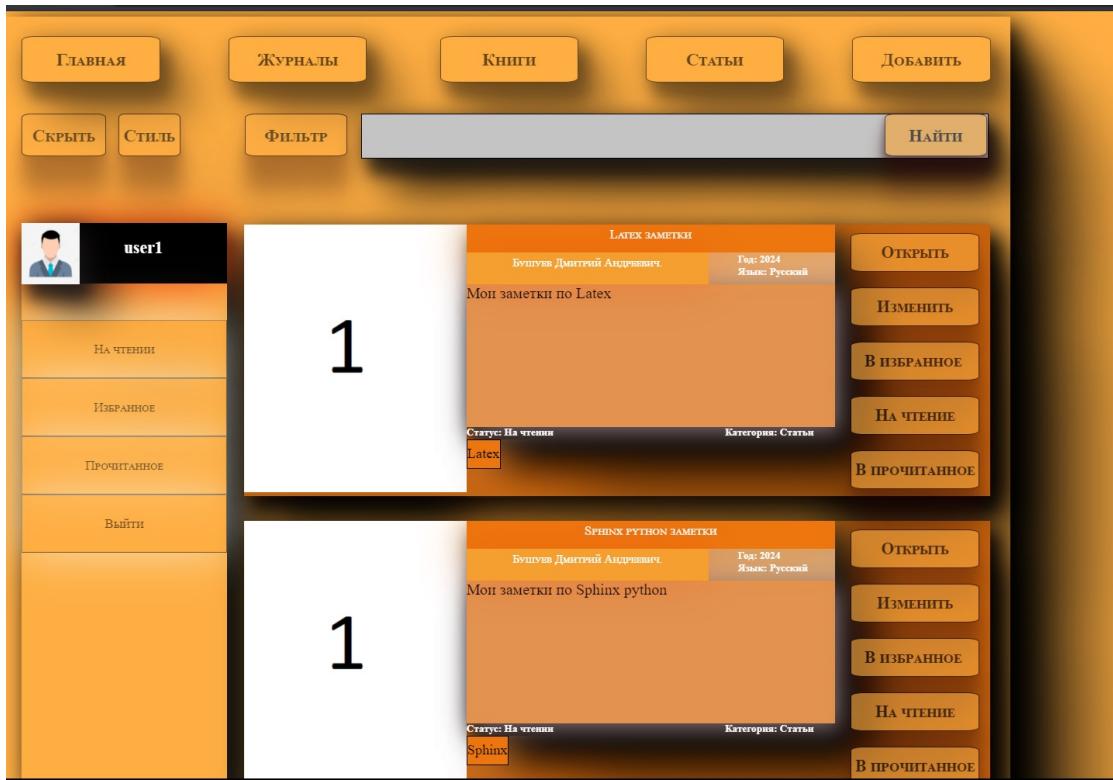
The screenshot shows the 'Добавление книги' (Add Book) page. On the left is a sidebar with a user profile icon and the name 'user1'. Below the profile are four buttons: 'На чтении', 'Изучение', 'Прочитанное', and 'Вычит'. The main content area is divided into three columns:

- Информация о книге**:
 - Название книги: input field
 - Количество страниц: input field
 - Путь к книге: input field containing '/static/books/'
 - Путь к изображению книги: input field containing '/static/images/books/'
 - Выберите категорию: dropdown menu
 - Выберите язык: dropdown menu
 - Выберите год: dropdown menu
- Авторы книги**:
 - Добавить нового автора: button
 - Фамилия: input field
 - Имя: input field
 - Отчество: input field
 - Добавить нового автора: button
 - Список авторов для книги: input field
 - Выберите автора: dropdown menu
 - Добавить автора в список: button
- Тэги книги**:
 - Добавить новый тэг: button
 - Название тэга: input field
 - Добавить новый тэг: button
 - Список тэгов для книги: input field
 - Выберите тэг: dropdown menu
 - Добавить тэг в список: button
 - Описание книги: input field
 - Добавить книгу: button

Страница добавления книги (стиль 5)



Страница с книгами (стиль 1)



Страница с книгами (стиль 2)

The screenshot displays a user interface for a library application. At the top, there is a navigation bar with five buttons: 'ГЛАВНАЯ' (Main), 'ЖУРНАЛЫ' (Journals), 'Книги' (Books), 'Статьи' (Articles), and 'Добавить' (Add). Below the navigation bar are several search and filter options: 'Скрыть' (Hide), 'Стиль' (Style), 'ФИЛЬТР' (Filter), and a large search input field with a 'Найти' (Find) button. On the left side, a sidebar for the user 'user1' shows links for 'На чтении' (Reading), 'Избранное' (Favorites), 'Прочитанное' (Read), and 'Выйти' (Logout).

The main content area shows two book cards:

LATEX ЗАМЕТКИ

- Буслев Дмитрий Андреевич
- Год: 2024
- Язык: Русский

Мои заметки по Latex

Статус: На чтении

Категория: Статья

Latex

SPHINX PYTHON ЗАМЕТКИ

- Буслев Дмитрий Андреевич
- Год: 2024
- Язык: Русский

Мои заметки по Sphinx python

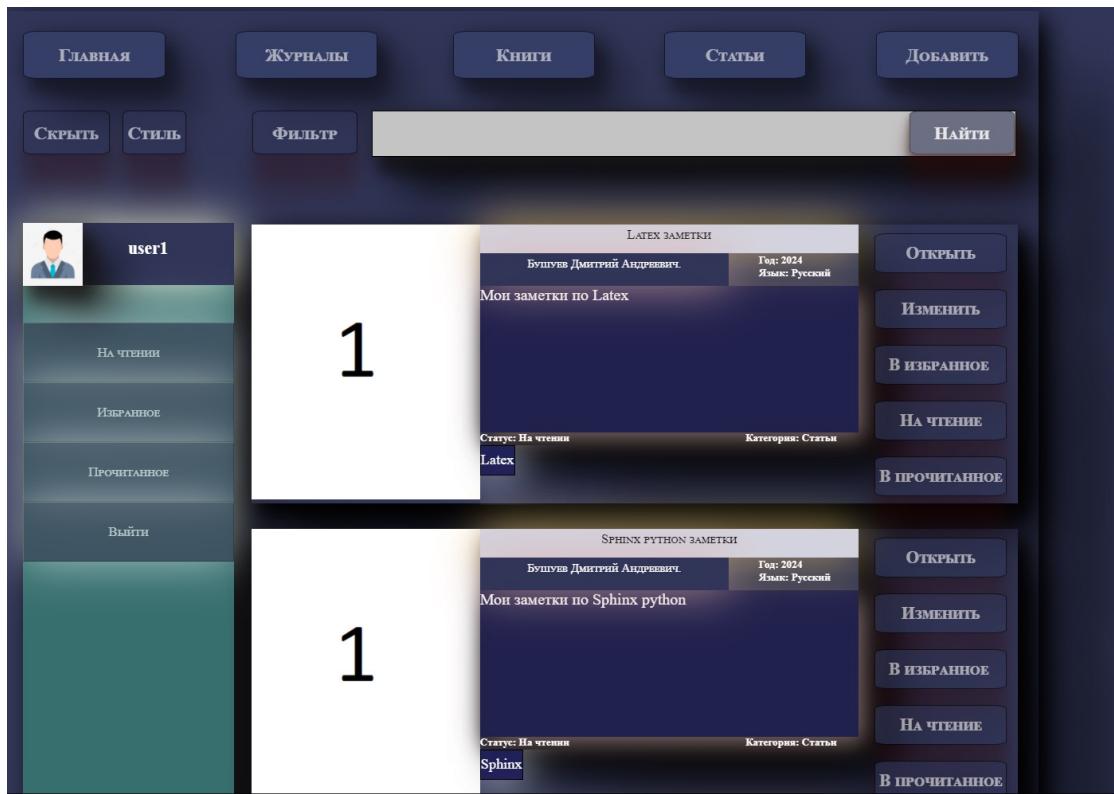
Статус: На чтении

Категория: Статья

Sphinx

On the right side of each card, there is a vertical column of buttons: 'Открыть' (Open), 'Изменить' (Edit), 'В избранное' (To Favorites), 'На чтение' (To Reading), and 'В прочитанное' (To Read).

Страница с книгами (стиль 3)



Страница с книгами (стиль 4)

The screenshot displays a library application interface with a dark green theme. At the top, there is a navigation bar with buttons for 'ГЛАВНАЯ' (Main), 'ЖУРНАЛЫ' (Journals), 'Книги' (Books), 'Статьи' (Articles), and 'Добавить' (Add). Below the navigation bar are buttons for 'Скрыть' (Hide), 'Стиль' (Style), 'ФИЛЬР' (Filter), and a search bar with a 'Найти' (Find) button. On the left, a sidebar for user 'user1' shows categories: 'На чтении' (Reading), 'Избранное' (Favorites), 'Прочитанное' (Read), and 'Выйти' (Logout). The main content area shows two book entries:

1

LATEX ЗАМЕТКИ
БУШУЕВ ДМИТРИЙ АНДРЕЕВИЧ Год: 2024
Мои заметки по Latex Язык: Русский
Статус: На чтении Категория: Статьи
Latex

1

SPHINX PYTHON ЗАМЕТКИ
БУШУЕВ ДМИТРИЙ АНДРЕЕВИЧ Год: 2024
Мои заметки по Sphinx python Язык: Русский
Статус: На чтении Категория: Статьи
Sphinx

On the right side of each entry, there are four buttons: 'Открыть' (Open), 'Изменить' (Edit), 'В избранное' (To Favorites), and 'На чтение' (To Reading). Below these, there are additional buttons: 'В прочитанное' (To Read), 'Открыть' (Open), 'Изменить' (Edit), 'В избранное' (To Favorites), 'На чтение' (To Reading), and 'В прочитанное' (To Read).

Страница с книгами (стиль 5)

The screenshot displays a dark-themed user interface for a library application. At the top, there is a navigation bar with five orange buttons labeled: ГЛАВНАЯ (Home), ЖУРНАЛЫ (Journals), КНИГИ (Books), СТАТЬИ (Articles), and ДОБАВИТЬ (Add). Below the navigation bar is a search bar with three buttons: Скрыть (Hide), Стиль (Style), and ФИЛЬТР (Filter). To the right of the search bar are two buttons: Найти (Find) and a large, empty input field.

On the left side, there is a sidebar for the user 'user1' with four categories: На чтении (Reading), Избранное (Favorites), Прочитанное (Read), and Выйти (Logout). The 'На чтении' category is currently selected.

The main content area shows two book cards:

- LATEX ЗАМЕТКИ**
Буслуков Дмитрий Андреевич | Год: 2024 | Язык: Русский
Мои заметки по Latex
Статус: На чтении | Категория: Статьи
Latex
- SPHINX PYTHON ЗАМЕТКИ**
Буслуков Дмитрий Андреевич | Год: 2024 | Язык: Русский
Мои заметки по Sphinx python
Статус: На чтении | Категория: Статьи
Sphinx

Each book card has a set of orange action buttons on the right: Открыть (Open), Изменить (Edit), В избранное (Add to Favorites), На чтение (Read), and В прочитанное (Add to Read).

Страница изменения информации о книге (стиль 1)

Информация о книге

Название книги
Latex заметки

Количество страниц
25

Путь к книге
/static/books/3.pdf

Путь к изображению книги
/static/images/books/1.jpg

Статьи

Русский

2024

Авторы книги

Добавить нового автора
Фамилия

Имя

Отчество

Добавить нового автора

Список авторов для книги

Выберите автора

Добавить автора в список

Тэги книги

Добавить новый тэг
Название тэга

Добавить новый тэг

Список тэгов для книги

Добавить

Выберите тэг

Добавить тэг в список

Описание книги

Мои заметки по Latex

Изменить

Страница изменения информации о книге (стиль 2)

Информация о книге

Название книги
Latex заметки

Количество страниц
25

Путь к книге
/static/books/3.pdf

Путь к изображению книги
/static/images/books/1.jpg

Статьи

Русский

2024

Авторы книги

Добавить нового автора
Фамилия

Имя

Отчество

Добавить нового автора

Список авторов для книги

Выберите автора

Добавить автора в список

Тэги книги

Добавить новый тэг
Название тэга

Добавить новый тэг

Список тэгов для книги

Добавить

Выберите тэг

Добавить тэг в список

Описание книги

Мои заметки по Latex

Изменить

Страница изменения информации о книге (стиль 3)

Страница изменения информации о книге (стиль 4)

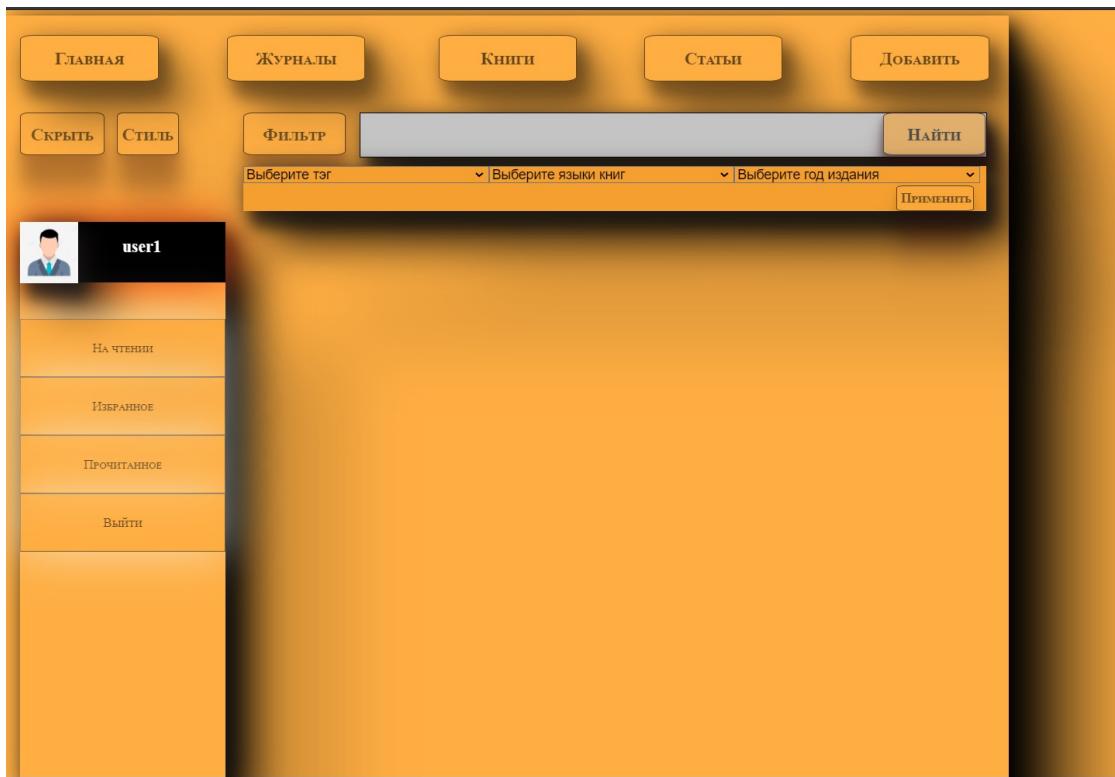
Страница изменения информации о книге (стиль 5)

The screenshot shows a dark-themed user interface for managing book information. At the top, there is a navigation bar with five buttons: Главная (Home), Журналы (Journals), Книги (Books), Статьи (Articles), and Добавить (Add). Below the navigation bar are two buttons: Скрыть (Hide) and Стиль (Style). A search bar labeled ФИЛЬТР (Filter) is followed by a search button Найти (Find).

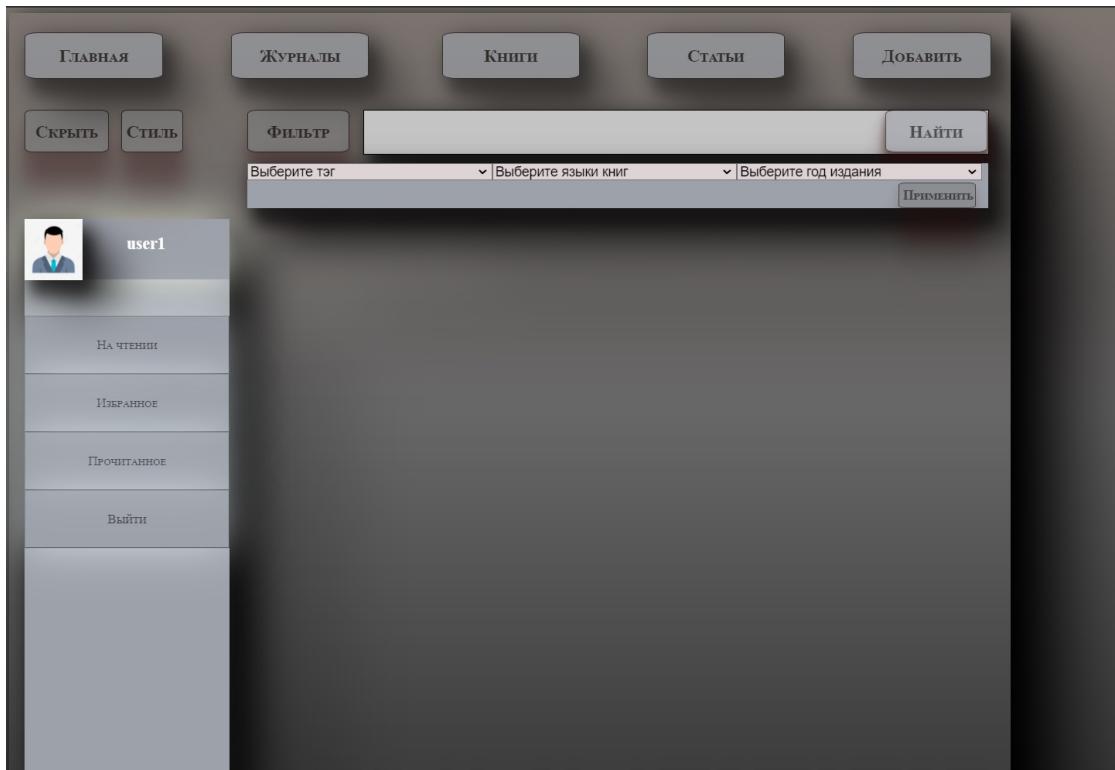
The main area contains several input fields and dropdown menus:

- Информация о книге (Information about the book):**
 - Название книги (Name of the book): Latex заметки (Latex notes). Includes an "Изменить" (Change) button.
 - Количество страниц (Number of pages): 25. Includes an "Изменить" (Change) button.
 - Путь к книге (Path to the book): /static/books/3.pdf. Includes an "Изменить" (Change) button.
 - Путь к изображению книги (Path to book image): /static/images/books/1.jpg. Includes an "Изменить" (Change) button.
 - Статьи (Category): Статьи. Includes an "Изменить" (Change) button.
 - Язык (Language): Русский. Includes an "Изменить" (Change) button.
 - Год (Year): 2024. Includes an "Изменить" (Change) button.
- Авторы книги (Authors of the book):**
 - Добавить нового автора (Add new author): Includes a "Фамилия" (Last name) input field and an "Изменить" (Change) button.
 - Имя (First name): Includes an "Изменить" (Change) button.
 - Отчество (Middle name): Includes an "Изменить" (Change) button.
 - Список авторов для книги (List of authors for the book): Includes a "Добавить нового автора" (Add new author) button and a "Добавить" (Add) button.
 - Выберите автора (Select author): Includes a dropdown menu and a "Добавить автора в список" (Add author to list) button.
- Тэги книги (Tags for the book):**
 - Добавить новый тэг (Add new tag): Includes a "Название тэга" (Tag name) input field and a "Добавить новый тэг" (Add new tag) button.
 - Список тэгов для книги (List of tags for the book): Includes a "Добавить" (Add) button.
 - Выберите тэг (Select tag): Includes a dropdown menu and a "Добавить тэг в список" (Add tag to list) button.
- Описание книги (Book description):**
 - Мои заметки по Latex (My notes on Latex). Includes an "Изменить" (Change) button.

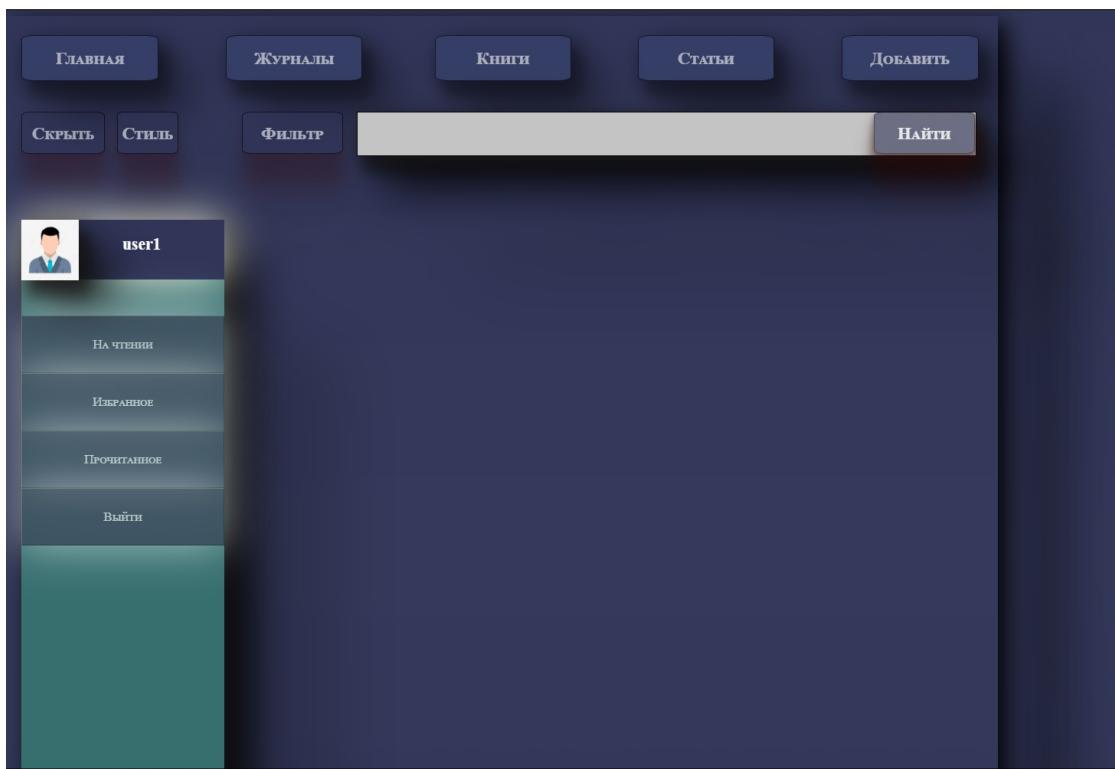
Окно фильтра (стиль 1)



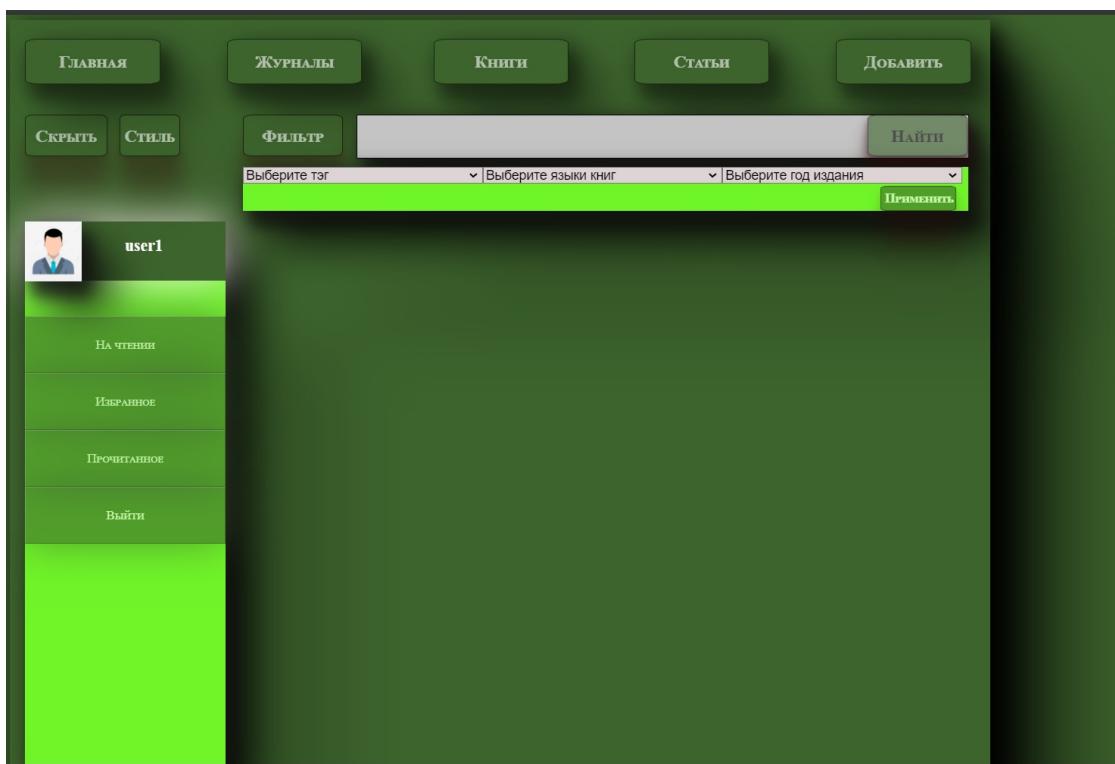
Окно фильтра (стиль 2)



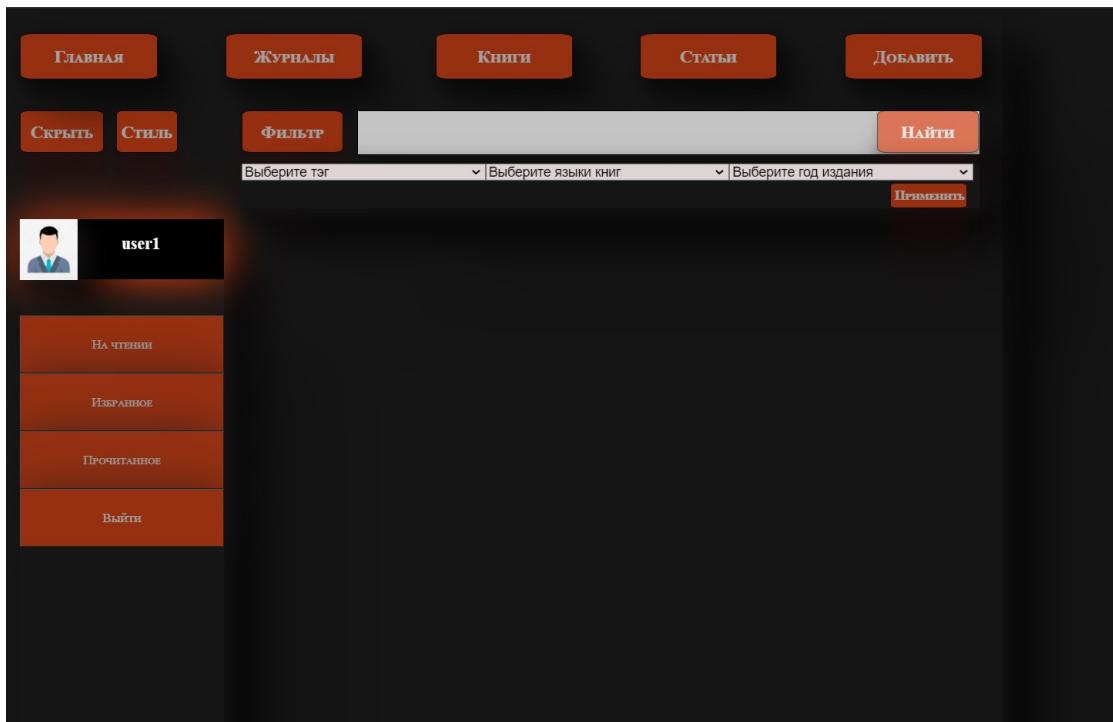
Окно фильтра (стиль 3)



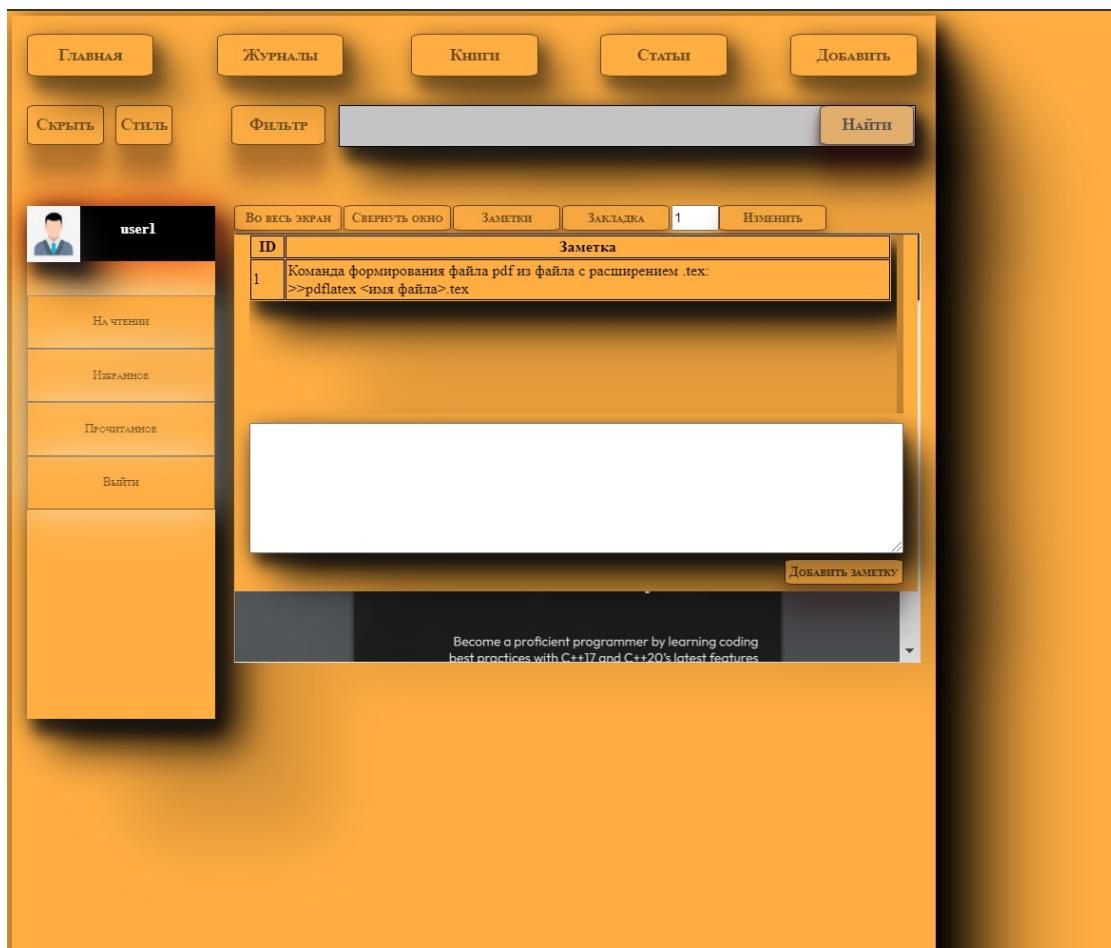
Окно фильтра (стиль 4)



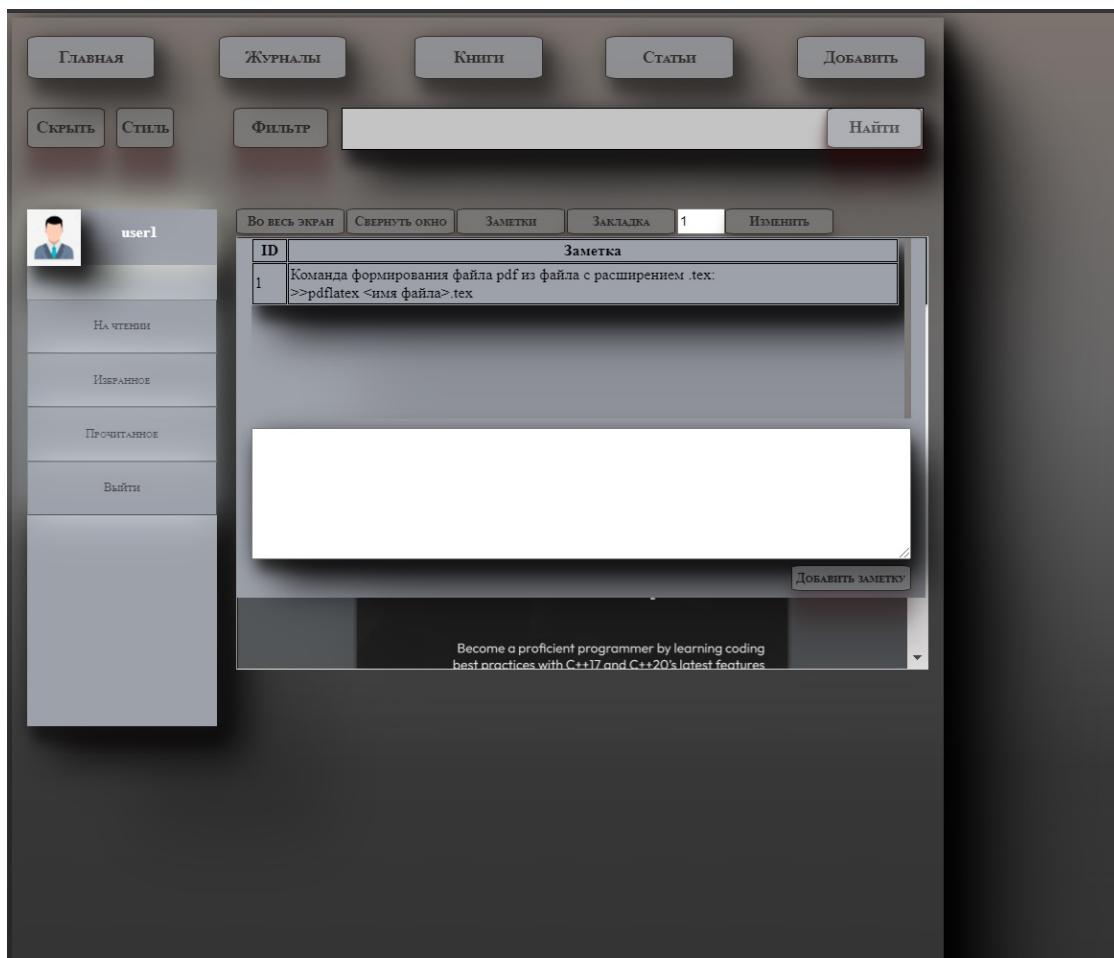
Окно фильтра (стиль 5)



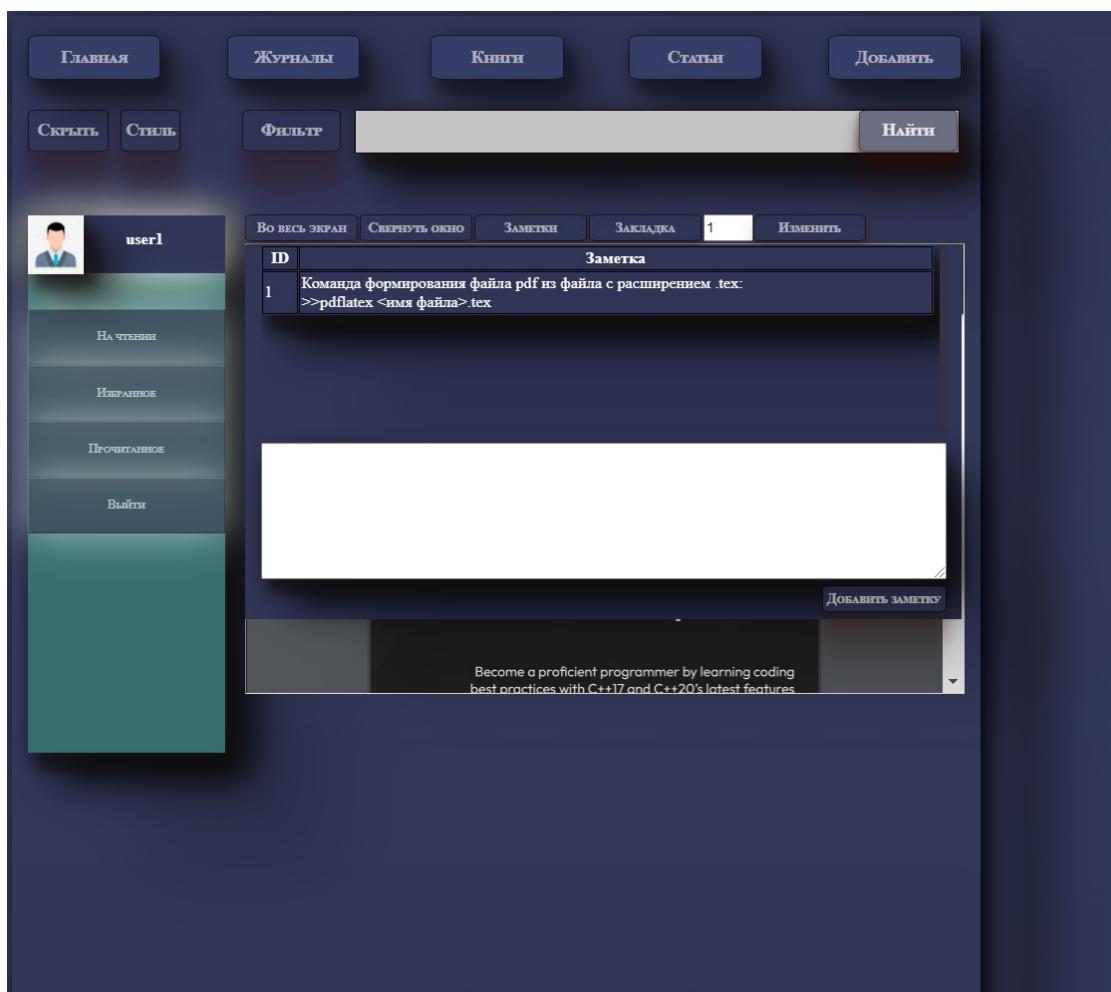
Окно заметок и закладки (стиль 1)



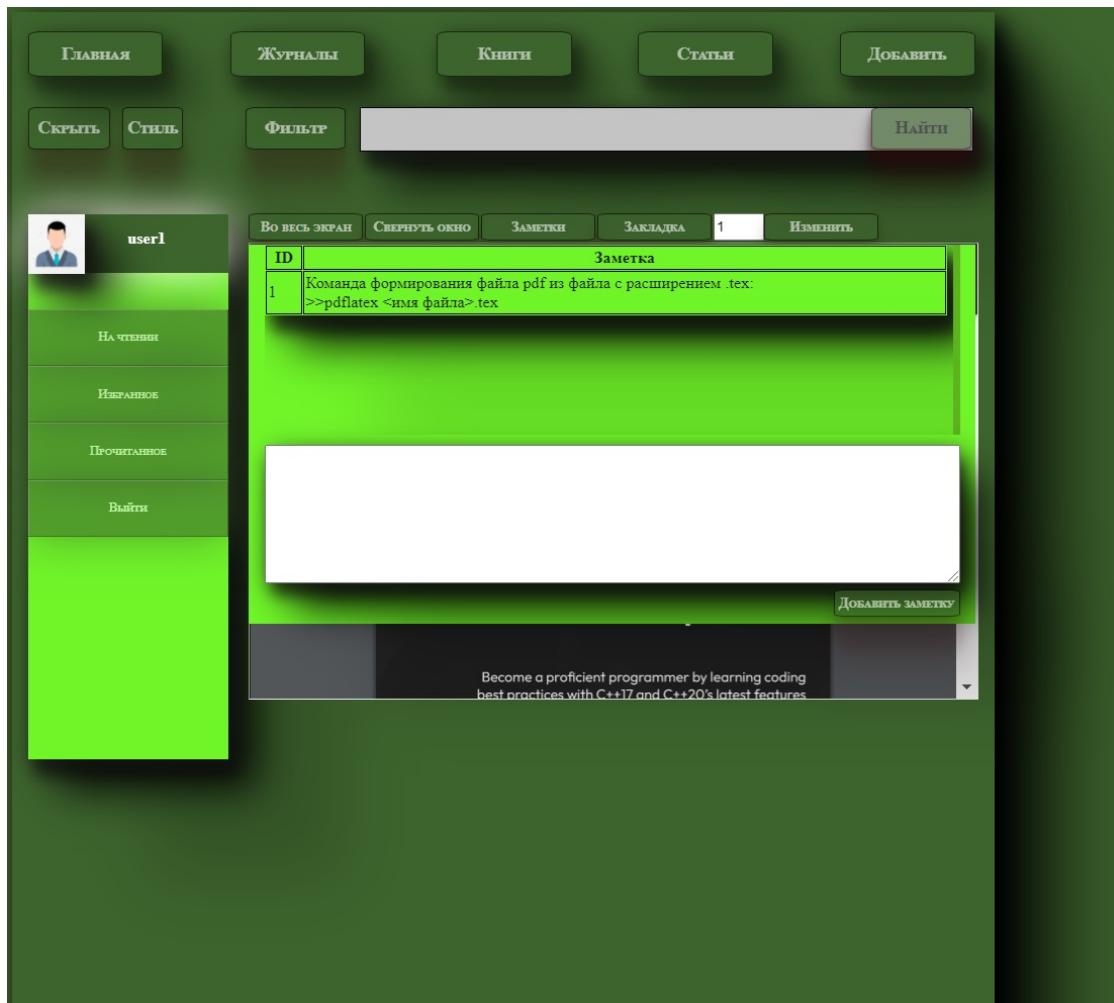
Окно заметок и закладки (стиль 2)



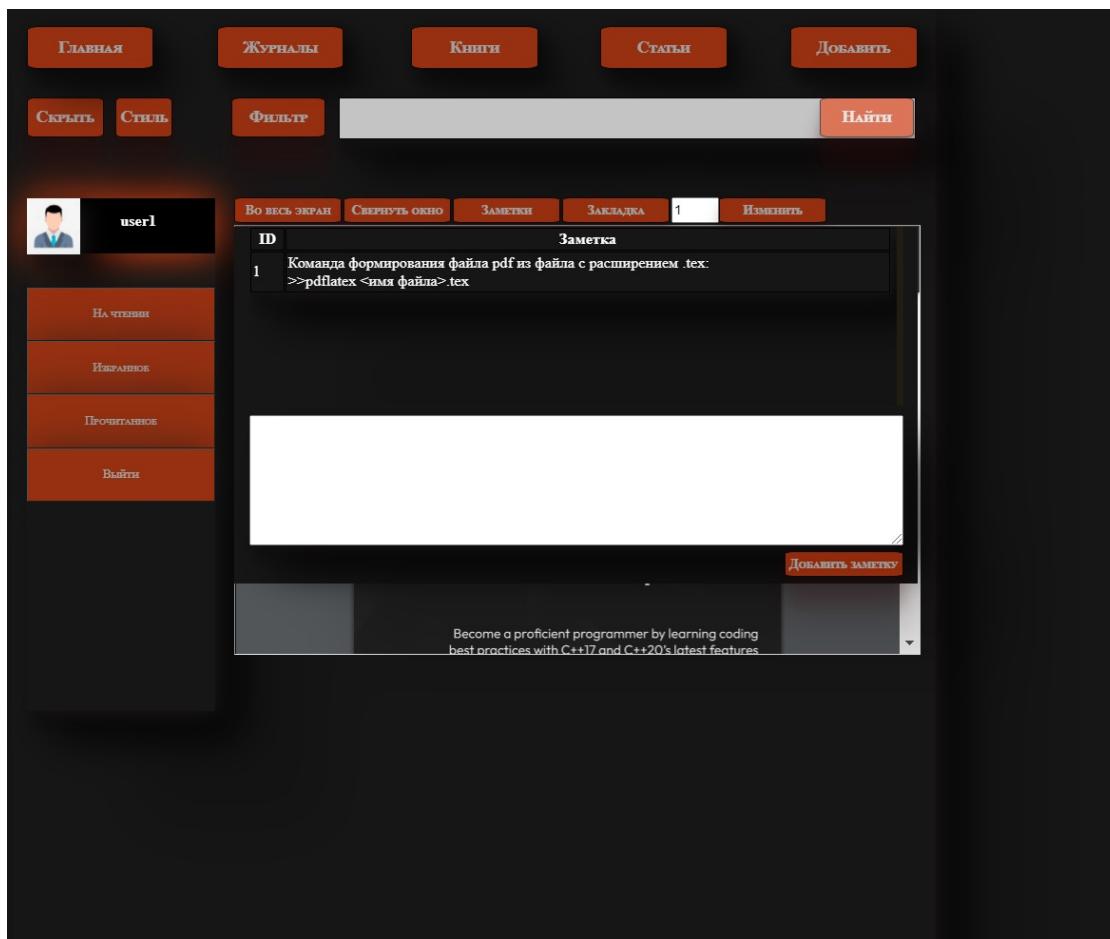
Окно заметок и закладки (стиль 3)



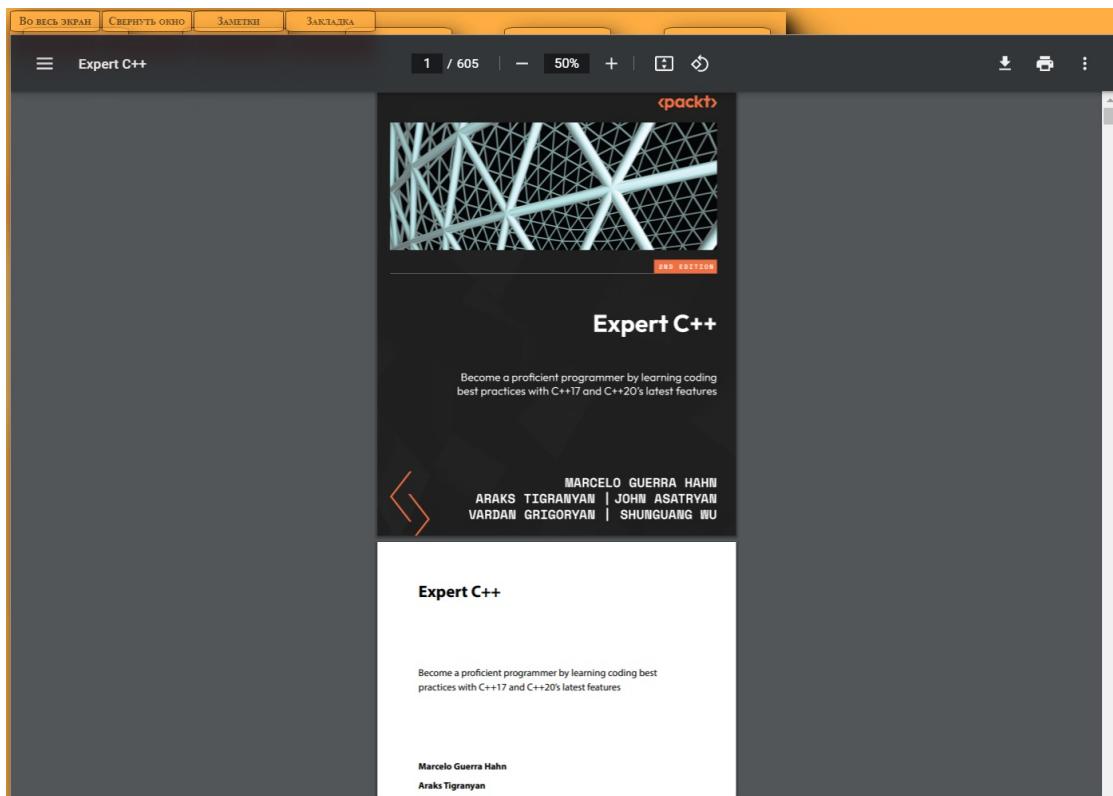
Окно заметок и закладки (стиль 4)



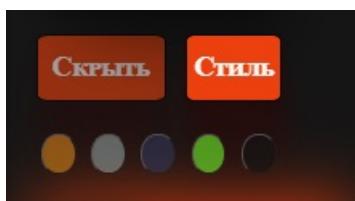
Окно заметок и закладки (стиль 5)



Окно чтения во весь экран



Меню изменения стиля



Содержание модулей Python

b
blueprints.add_to_database, 41
blueprints.auth, 44
blueprints.book_read, 45
blueprints.change_book_information, 47
blueprints.change_state, 50
blueprints.function, 51
blueprints.search_filter, 52
blueprints.select_books, 55
blueprints.select_books_state, 58
blueprints.style, 61

d
database.sqlite3_interface.create_db, 37
database.sqlite3_interface.tables.author, 6
database.sqlite3_interface.tables.book, 8
database.sqlite3_interface.tables.category,
 12
database.sqlite3_interface.tables.language,
 13
database.sqlite3_interface.tables.list_authors,
 14
database.sqlite3_interface.tables.list_notes,
 15
database.sqlite3_interface.tables.list_states,
 16
database.sqlite3_interface.tables.list_tags,
 19
database.sqlite3_interface.tables.note, 20
database.sqlite3_interface.tables.state, 22
database.sqlite3_interface.tables.table, 5
database.sqlite3_interface.tables.tag, 23
database.sqlite3_interface.tables.user, 24
database.sqlite3_interface.tables.year, 26
database.sqlite3_interface.views.view, 27
database.sqlite3_interface.views.view_articles,
 27
database.sqlite3_interface.views.view_books,
 28

database.sqlite3_interface.views.view_end_read,
 29
database.sqlite3_interface.views.viewFavorites,
 30
database.sqlite3_interface.views.view_filter,
 31
database.sqlite3_interface.views.view_in_read,
 34
database.sqlite3_interface.views.view_journals,
 35
database.sqlite3_interface.views.view_search,
 36