*Московский государственный технический университет*

*имени Н. Э. Баумана*

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

## ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ

**по курсу «Тестирование и отладка ПО»**

Студенты ИУ7-75: Марабян К.В.

Селивёрстов И.А.

Маркарьян К.Х.

Преподаватель: Рогозин Н. О.

Москва 2019

**Цель лабораторной работы**

Написать спецификацию программного продукта и протестировать его с помощью модульных, интеграционных, регрессионных, функциональных и системных тестов.

**Задача**

Целью работы является реализация веб-приложения для моделирования работы сайта библиотеки. Необходимо обеспечить хранение данных о книгах, авторах и пользователях. Подключение новых пользователей происходит с помощью регистрации – создания новой записи в базе данных. После аутентификации пользователь может воспользоваться всем функционалом сайта.

Данное WEB-приложение написано на языке Python с использованием фреймворка Django, который основан на концепции MVC(Model-View-Controller). Также была использована СУБД SQLite.

# Модульное тестирование

Цель модульного тестирования — изолировать отдельные части программы и показать, что по отдельности эти части работоспособны. С помощью модульных тестов мы протестировали основные компоненты нашего приложения.

Модульное тестирование производилось на методом registration()

Соответствующие классы эквивалентности registration():

|  |
| --- |
| def test\_registration\_newUser(self):  resp = self.client.get('/library/registration/')  resp = self.client.post(reverse('registration'), {'username': 'Koryun', 'password': '11091998k', 'email': '[marabyankoi@gmail.com](mailto:marabyankoi@gmail.com)', 'first\_name': 'Koryun', 'last\_name': 'Marabyan'})  self.assertEqual(resp.status\_code, 200)  self.assertRedirects(resp, '/accounts/login/') |

Добавление пользователя:

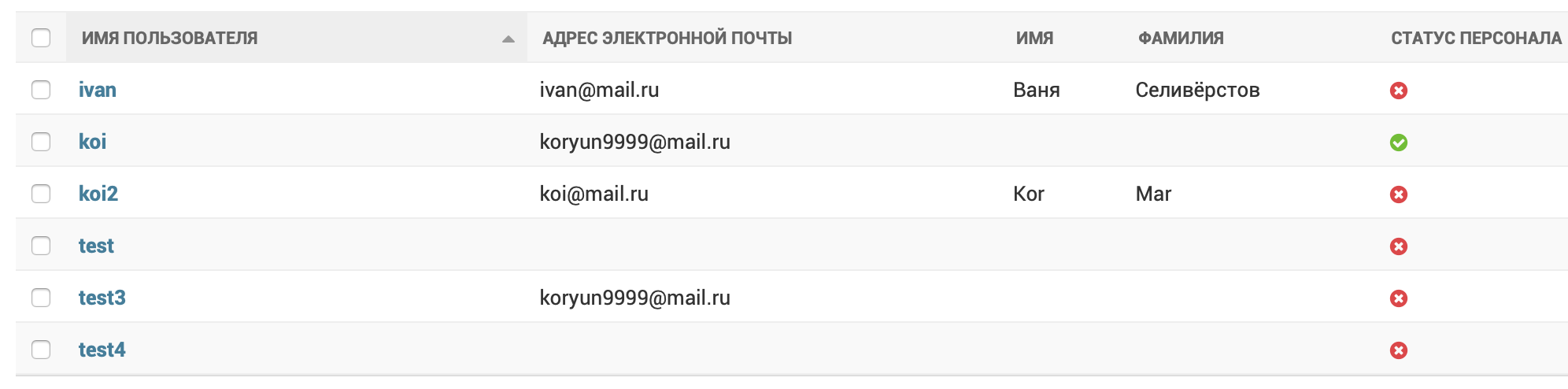
- добавить нового пользователя с не существующими в базе данных email и login

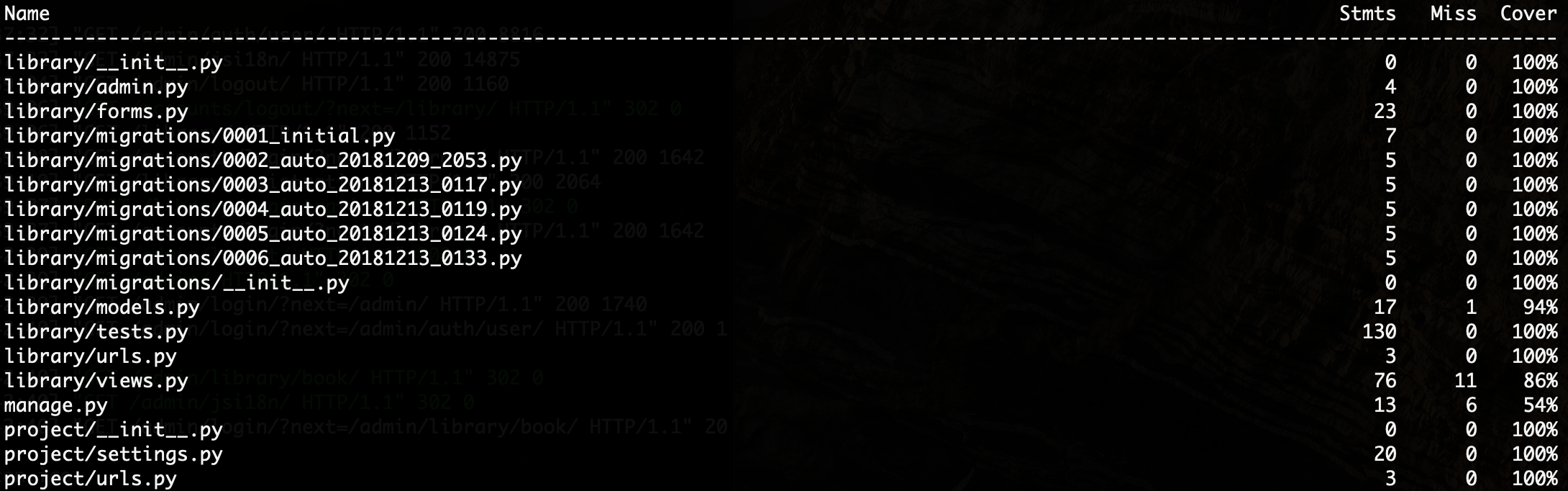
- добавить пользователя с существующим логином

- добавить пользователя с существующим email

- добавить пользователя не заполнив все поля

|  |  |
| --- | --- |
| **Класс** | **Результат** |
| Значение полей:  Login - test  Password - t123456  Name - MyName  Surname - MySurname  eMail - [test@mail.ru](mailto:test@mail.ru) | Success!  Redirect to the main page! |
| Значение полей:  Login - test  Password - t123456  Name - MyName  Surname - MySurname  eMail - [test@mail.ru](mailto:test@mail.ru) | Error!  «This login does exist! Please try again!»  Redirect to the main page! |
| Значение полей:  Login - test2  Password - t123456  Name - MyName  Surname - MySurname  eMail - [test@mail.ru](mailto:test@mail.ru) | Success!  Redirect to the main page! |
| Значение полей:  Login - test3  Password - t123456  Name -  Surname -  eMail - | Success!  Redirect to the main page! |
| Значение полей:  Login -  Password - t123456  Name - MyName  Surname - MySurname  eMail - [test@mail.ru](mailto:test@mail.ru) | Error!  «Please, enter your login!» |

База данных:

Покрытие кода тестами:

# Интеграционное тестирование

Целью интеграционного тестирования является проверка соответствия проектируемых единиц функциональным, приёмным и требованиям надежности. Тестирование этих проектируемых единиц — объединения, множества или группы модулей — выполняется через их интерфейс, с использованием тестирования «чёрного ящика».

Интеграционные тесты похожи на модульные тесты, но их отличие состоит в том, что тестируется совместно работа базы данных и тестируемых классов.

Интеграционное тестирование производилось на методоми LibraryTest() и registration()

Соответствующие классы эквивалентности LibraryTest():

|  |
| --- |
| def test\_search\_book\_redirect\_if\_not\_logged\_in(self):  resp = self.client.get('/library/search\_book/')  self.assertRedirects(resp, '/accounts/login/?next=/library/search\_book/')  def test\_search\_book\_logged\_in\_correct\_get\_request(self):  login = self.client.login(username='koi', password='11091998k')  resp = self.client.get('/library/search\_book/')  self.assertEqual(str(resp.context['user']), 'koi')  self.assertEqual(resp.status\_code, 200)  self.assertTemplateUsed(resp, 'library/input.html') |

Поиск книги:

* найти книгу со своего аккаунта
* найти книгу без входа в аккаунт

Соответствующие классы эквивалентности registration():

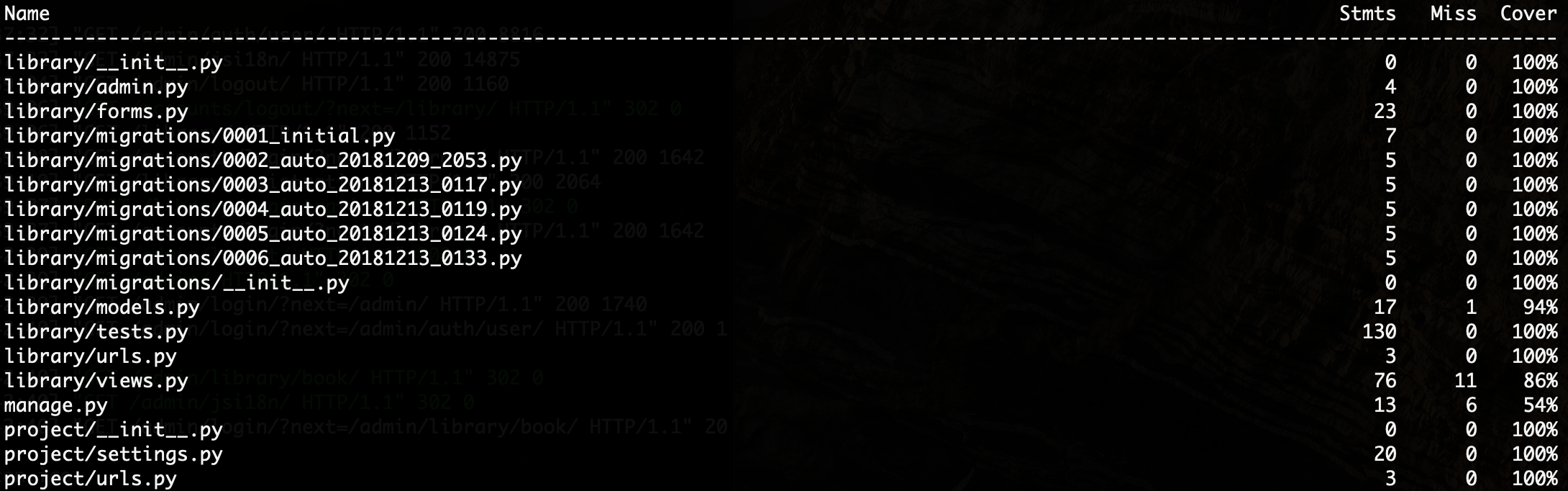
Добавление пользователя:

- добавить нового пользователя с не существующими в базе данных email и login

- добавить пользователя с существующим логином

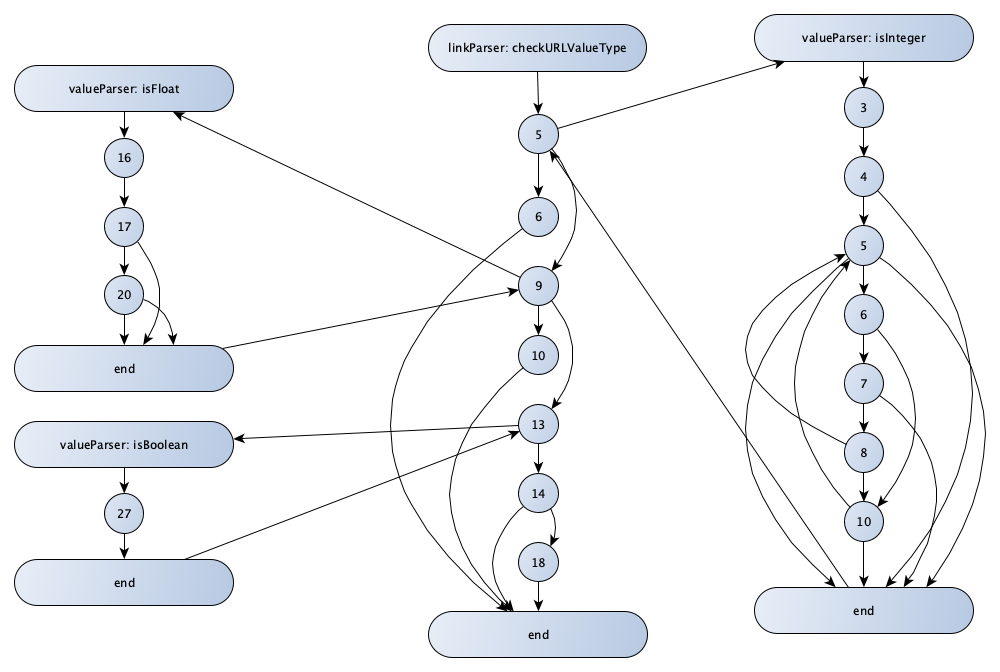
* добавить пользователя с существующим email
* добавить пользователя не заполнив все поля

|  |  |
| --- | --- |
| **Класс** | **Результат** |
| search\_book\_redirect\_if\_not\_logged\_in() | Error!  Redirect to the main page! |
| search\_book\_logged\_in\_correct\_get\_request() | Success! |
| Значение полей:  Login - test  Password - t123456  Name - MyName  Surname - MySurname  eMail - [test@mail.ru](mailto:test@mail.ru) | Success!  Redirect to the main page! |
| Значение полей:  Login - test  Password - t123456  Name - MyName  Surname - MySurname  eMail - [test@mail.ru](mailto:test@mail.ru) | Error!  «This login does exist! Please try again!»  Redirect to the main page! |
| Значение полей:  Login - test2  Password - t123456  Name - MyName  Surname - MySurname  eMail - [test@mail.ru](mailto:test@mail.ru) | Success!  Redirect to the main page! |
| Значение полей:  Login - test3  Password - t123456  Name -  Surname -  eMail - | Success!  Redirect to the main page! |
| Значение полей:  Login -  Password - t123456  Name - MyName  Surname - MySurname  eMail - [test@mail.ru](mailto:test@mail.ru) | Error!  «Please, enter your login!» |

Покрытие кода тестами:**

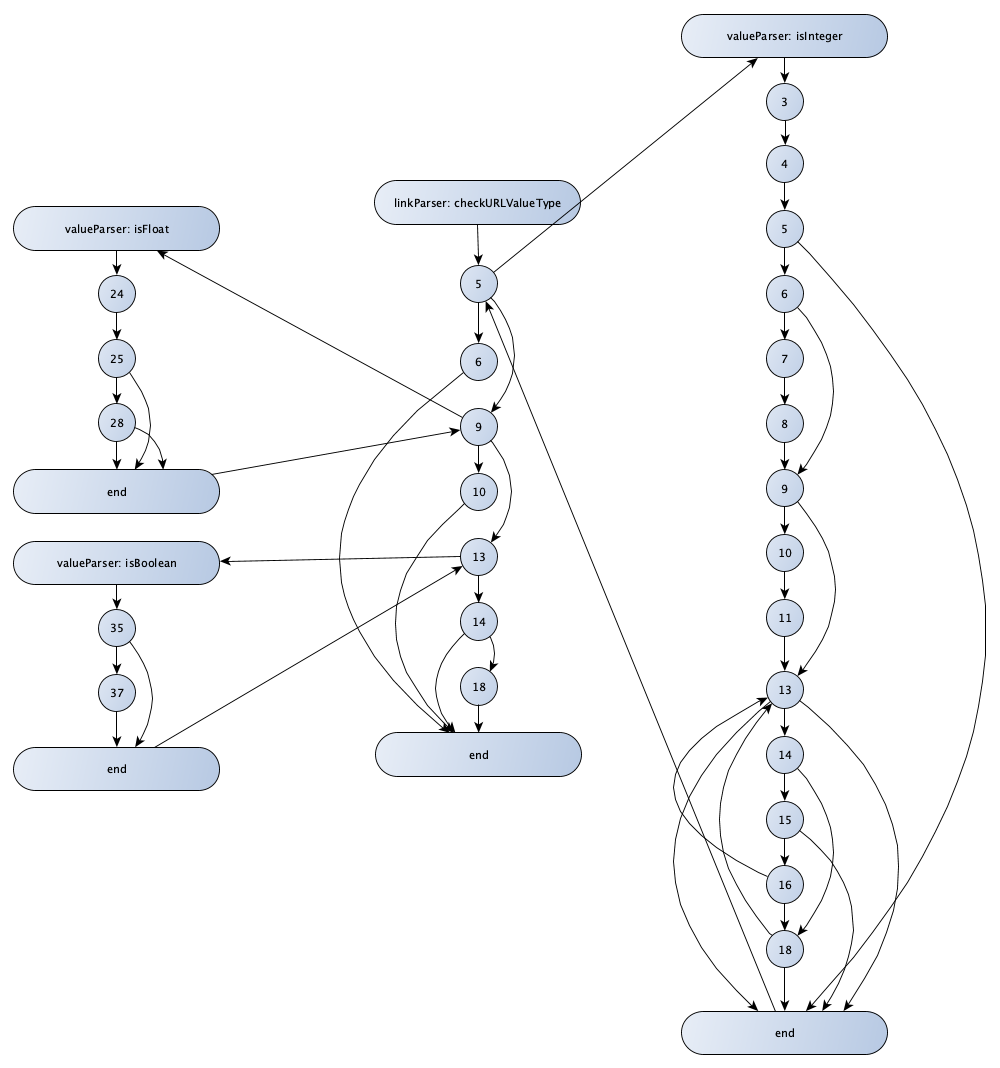
# Регрессионное тестирование

В ходе реализации программы было решено добавить возможность искать авторов книг и книги без входа в учетную запись. Так же при регистрации нового пользователя поля, которые были не обязательны для заполнения, теперь стали обязательными.

*Control flow graph до изменений в коде:*

*Новые тесты:*

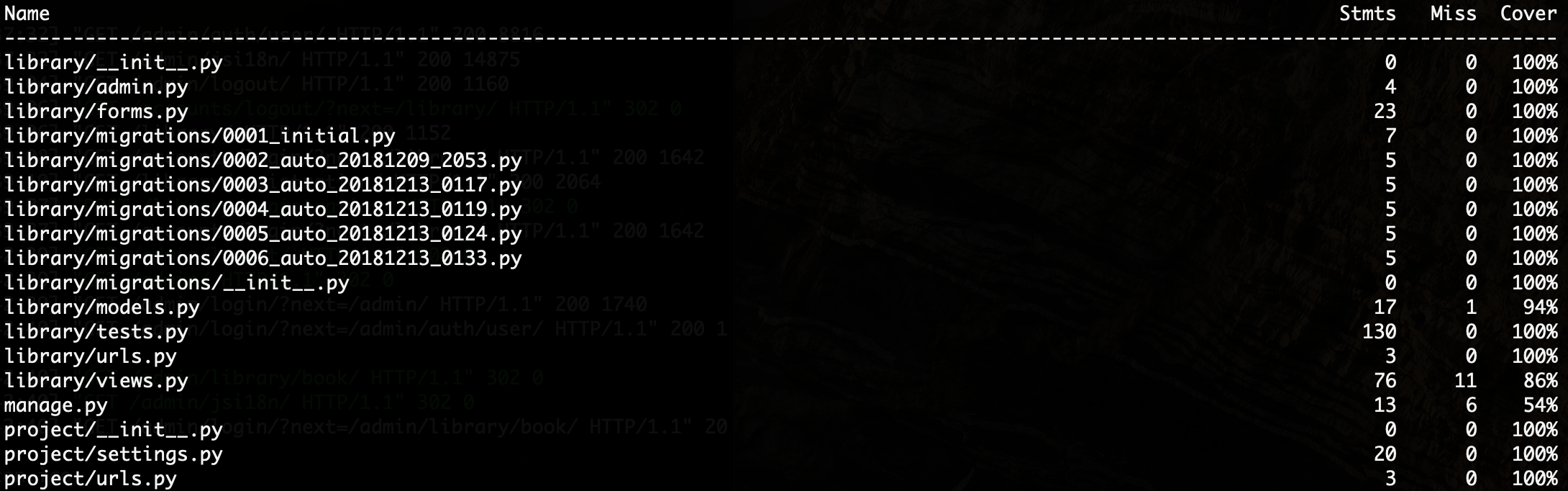
|  |  |
| --- | --- |
| **Входной**  **URLParam** | **Результат** |
| ?testparam=0x15a | URLParam.type.INTEGER |
| ?testparam=0x15gf | URLParam.type.STRING |
| ?testparam=True | URLParam.type.BOOLEAN |
| ?testparam=False | URLParam.type.BOOLEAN |
| ?testparam=035 | URLParam.type.INTEGER |
| ?testparam=09 | URLParam.type.STRING |
| ?testparam=.5 | URLParam.type.FLOAT |

*Control flow graph после изменений в коде:*

# Функциональное тестирование

Функциональное тестирование производилось методами search\_author(), search\_book(), search\_date(). Данные методы позволяют зарегистрированному пользователю искать в библиотеке Автора по его фамилии, книгу по её названию и книгу по дате её издания.

|  |  |
| --- | --- |
| **Входная строка** | **Результат** |
| search\_author()  Фамилия автора - | «Заполните поле!» |
| search\_author()  Фамилия автора - Селиверстов | Появится список книг написанных данным автором |
| search\_author()  Фамилия автора - Маркарьян | Если книги с указанным автором нет в библиотеке, то выдаст сообщение «Книги с таким автором нет» |
| search\_book()  Название книги - | «Заполните поле!» |
| search\_book()  Название книги - МГТУ | Появится указанная книга с ее полным описанием |
| search\_book()  Название книги - МГУ | Если книги с указанным названием нет в библиотеке, то выдаст сообщение «Книги с таким названием нет» |
| search\_date()  Дата издания - | «Заполните поле!» |
| search\_date() - 2019-11-27 | Появятся книги(если они есть), с указанной датой издания |
| search\_date() - 2018-10-9 | Если книги с указанной датой нет в библиотеке, то выдаст сообщение «Книги с данной датой издания нет» |

Покрытие кода тестами:

# Многопоточное тестирование

Django не поддерживает несколько потоков одновременно, поэтому для поддержки многопоточности был скачен nginx. В конфигурационный файл nginx.confдобавлен следующий код:

server {  
 server\_name 127.0.0.1;  
 listen 8080;  
 location / {  
 proxy\_pass http://127.0.0.1:8000/;  
}

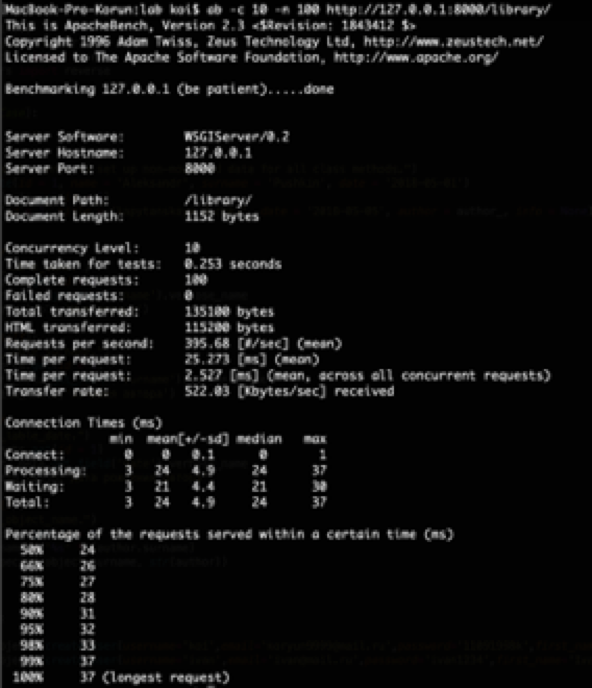
Для тестирования производительности сайта использовалась утилита Apache Benchmark.

Команда:

ab –c 10 –n 100 http://127.0.0.1:8080/library/

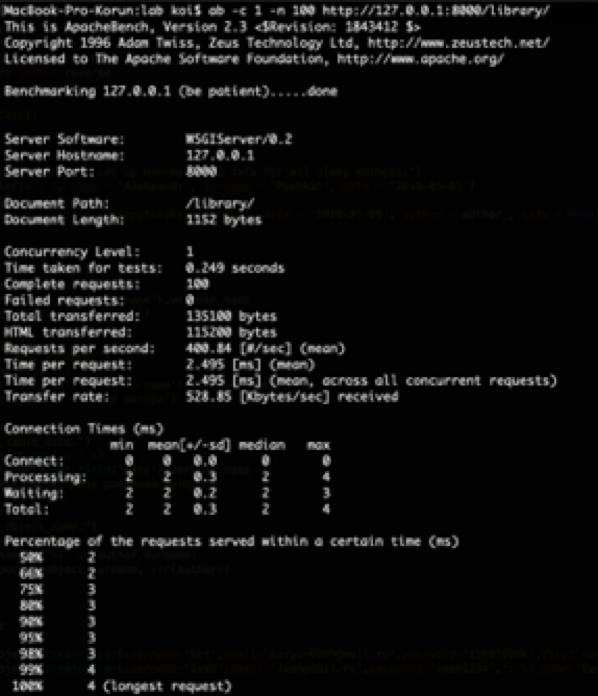
c = 10 – количество конкурентных запросов

n = 100 – общее количество запросов

Результат:

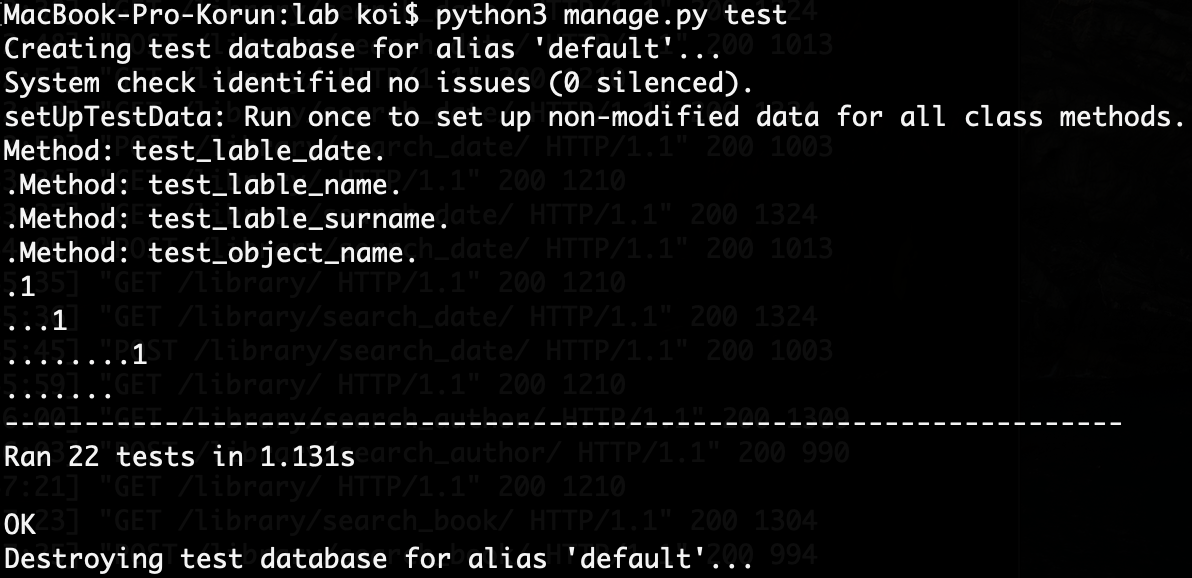
Команда:

ab –c 1 –n 100 http://127.0.0.1:8080/library/

Результат:

# Автоматизированное тестирование

Был написан скрипт для запуска тестов в соответствии с получаемым списком параметров.

*Проход тестов:*