Министерство науки и высшего образования РФ

ФГБОУ ВО Югорский государственный университет

|  |
| --- |
| ОТЧЕТ О КУРСОВОЙ РАБОТЕ  по теме:    «АГРЕГАТОР БИБЛИОТЕК ГОРОДА» |
|  |
|  |
| Студент гр. 1572б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Д.А. Парунов/  (подпись) |
| Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Р.Т. Усманов/  (подпись) |
|  |
|  |
| Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |

г. Ханты-Мансийск, 2020

Оглавление

[Введение 3](#_Toc62973843)

[Основная часть 4](#_Toc62973844)

[Цели и задачи 4](#_Toc62973845)

[1. DFD диаграмма 5](#_Toc62973846)

[2. ER модель 7](#_Toc62973847)

[3. Диаграмма прецедентов 8](#_Toc62973848)

[4. Диаграмма классов 10](#_Toc62973849)

[5. Реализация 12](#_Toc62973850)

[5.1.Реализация регистрации и авторизации аккаунта 16](#_Toc62973851)

[5.2.Реализация закрепления книги 20](#_Toc62973852)

[5.3.Реализация страницы «Список книг» 21](#_Toc62973853)

[6. Тестирование 22](#_Toc62973854)

[6.1.Тестирование регистрации нового пользователя и добавления его в базу данных. 23](#_Toc62973855)

[6.2. Тестирование добавления книги в базу данных 28](#_Toc62973856)

[6.3.Тестирование прикрепления книги к пользователю 30](#_Toc62973857)

[Заключение 32](#_Toc62973858)

# Введение

В современном мире все реже используется бумажные копии книг. Цены на книги с каждым годом все больше поднимаются, а иногда необходимая литература перестает публиковаться. Однако есть библиотеки, в которых можно найти нужную книги и взять ее бумажную копии в личное пользование на определенный срок. Однако не всегда известно, где находится библиотека, и какая литература в ней находится. Поэтому для более удобного поиска библиотек и литературы можно использовать сайты агрегаторы библиотек.

В данном курсовом проекте описывается создание сайта агрегатора библиотек города, включая DFD диаграмму , ER модель, диаграмму прецедентов, диаграмму классов, реализацию, тестирование, заключение.

# Основная часть

## Цели и задачи

Цель:

Целью данного проекта является создание сайта агрегатора библиотек города, при помощи которого можно найти и взять в личное пользование бумажную копию книги на определенный срок.

Задачи:

* Реализовать регистрацию и авторизацию учетной записи;
* Создать базу данных;
* Реализовать личный кабинет пользователя;
* Реализовать страницу списка книг;
* Реализовать закрепление книги;
* Реализовать страницу добавления книги.

## DFD диаграмма

Данная диаграмма описывает один из элементов сайта: процесс создания аккаунта. Диаграмма представлена на рис.1.1 и рис.1.2.

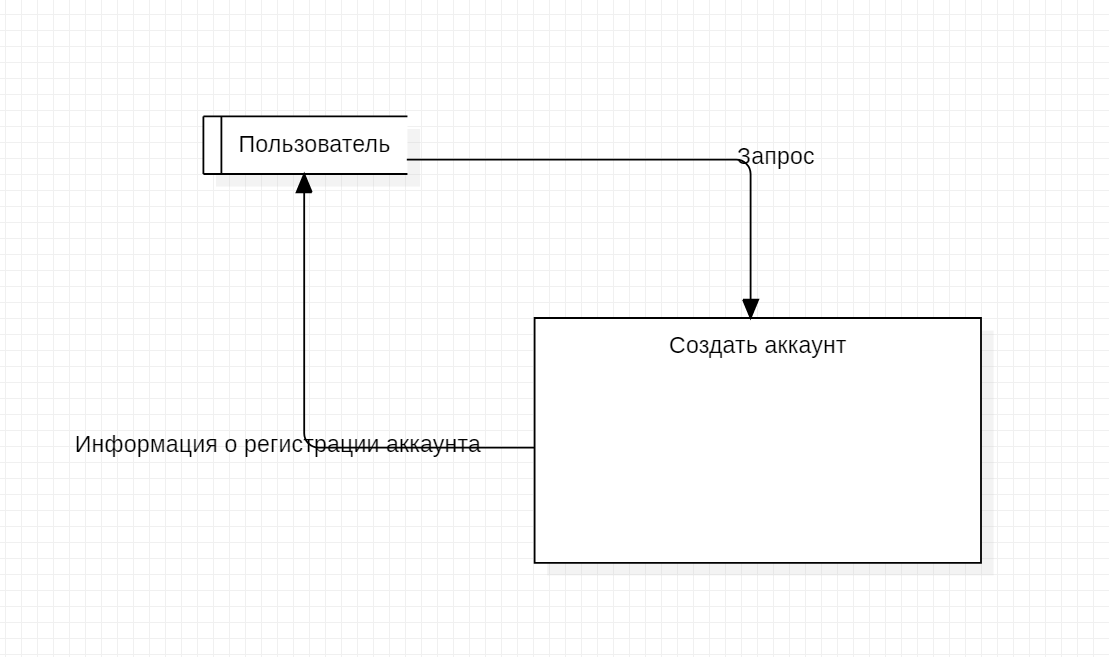


Рис.1.1 – Диаграмма DFD процесса создания аккаунта(1)

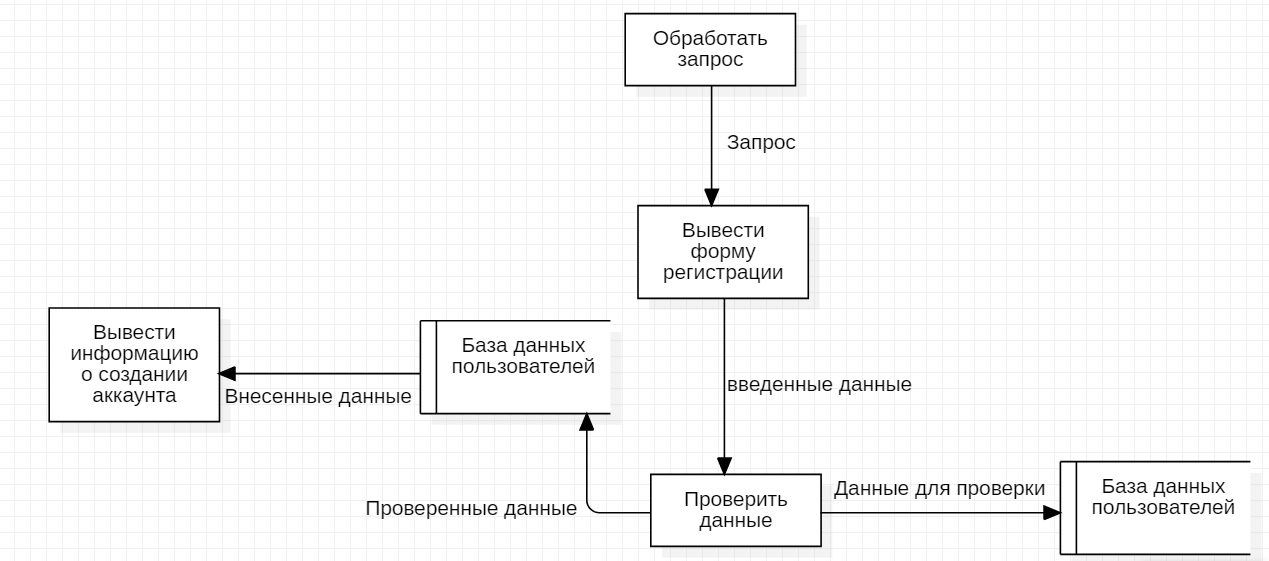


Рис.1.2 – Диаграмма DFD процесса создания аккаунта(2)

## ER модель

Для операции закрепления книги была создана ER модель. В ней присутствуют следующие сущности: Пользователь, форма закрепления, детали формы закрепления, книга, библиотека, детали библиотеки. ER модель представлена на рис.2.1.

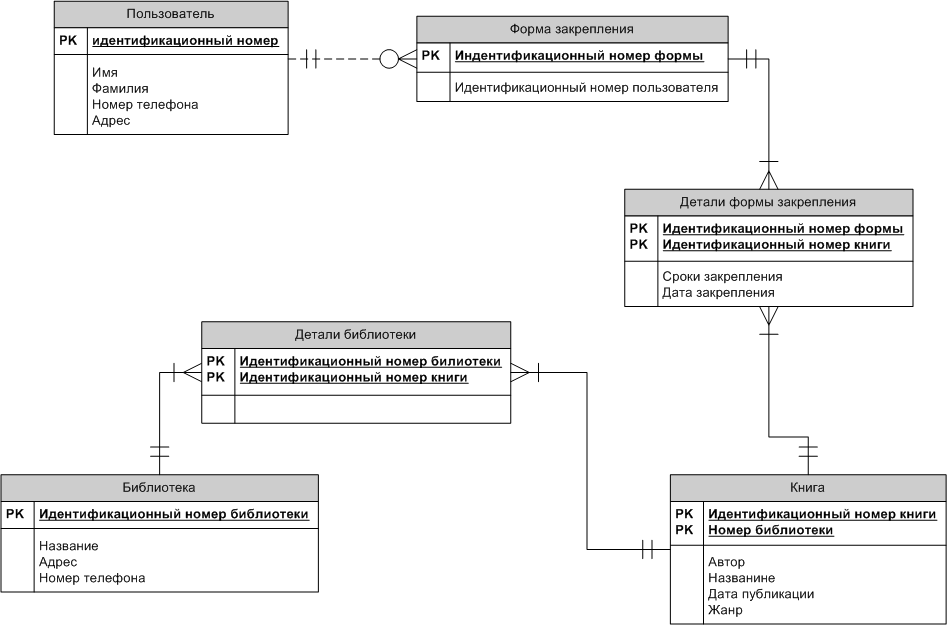


Рис.2.1 – ER модель операции прикрепления книги

## Диаграмма прецедентов

Для данного проекта была создана диаграмма прецедентов, которая описывает основные действия на сайте. Она состоит из следующих акторов:

* Пользователь;
* Система;
* Библиотека.

Включает следующие прецеденты:

* Открытие сайта;
* Авторизация или регистрация;
* Подтверждение учетной записи;
* Открытие страницы списка книг;
* Вывод списка книг;
* Выбор книги;
* Вывод информации о книге;
* Изменение информации о книге;
* Прикрепление книги;
* Редактирование списка книг;
* Редактирование списка пользователей.

Диаграмма прецедентов представлена на рис.3.1.

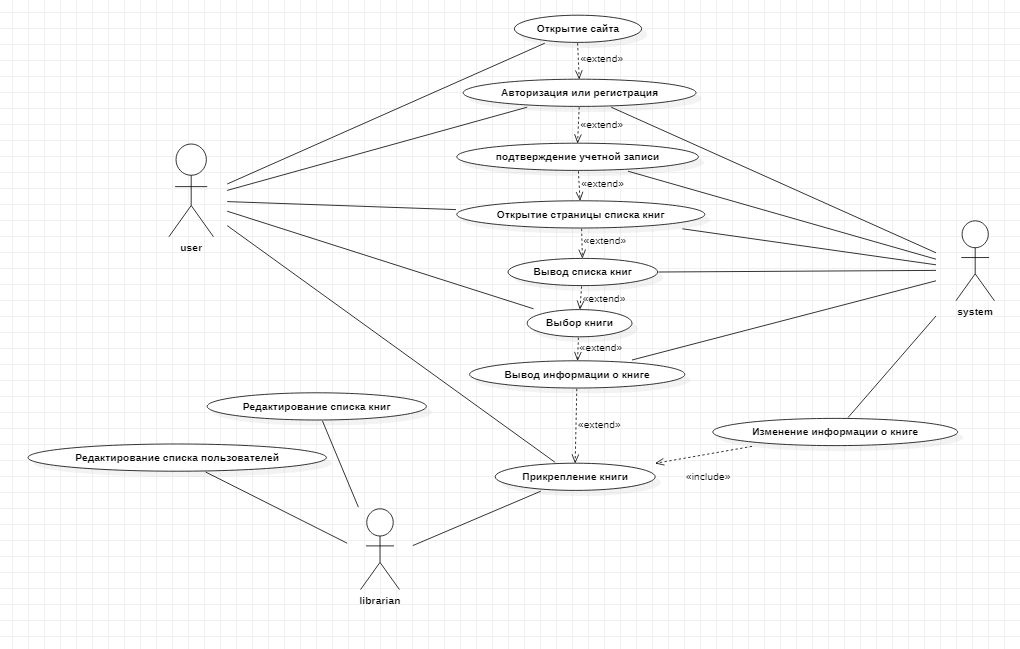


Рис.3.1 – Диаграмма прецедентов

## Диаграмма классов

Диаграмма классов демонстрирует классы, из которых состоит проект и взаимосвязь между ними. В диаграмму входят следующие классы:

* Система;
* База данных;
* Пользователь;
* Библиотекарь.

Класс «Система» включает в себя следующие операции:

* Регистрация учетной записи;
* Авторизация учетной записи;
* Прикрепление книги;
* Изменение базы данных.

Класс «База данных» состоит следующих атрибутов:

* Данные о книгах;
* Данные о библиотеках;
* Идентификационные номера пользователей;
* Идентификационные номера книг.

Класс «Пользователь» состоит из следующих атрибутов и операций:

Атрибуты:

* ФИО;
* Номер телефона;
* Адрес;
* Электронная почта.

Операции:

* Введение необходимых данных;
* Выбор книги.

Класс «Библиотекарь» состоит из следующих атрибутов и операций:

Атрибуты:

* ФИО;
* Должность.

Операции:

* Выдача книги;
* Редактирование базы данных;
* Прием сданной книги.

Диаграмма классов представлена на Рис.4.1.



Рис.4.1 – Диаграмма классов

## Реализация

Языком программирования был выбран Python с использованием редактора кода PyCharm. В качестве СУБД был использован SQLite.

В результате разработки был создан сайт и база данных. Были реализованы дизайн и следующие страницы сайта: главная страница (рис.5.1), список литературы (рис.5.2), список библиотек (рис.5.3), личный кабинет (рис.5.4), подробное описание книги (рис.5.5), страница регистрации (5.6), страница авторизации (5.7).

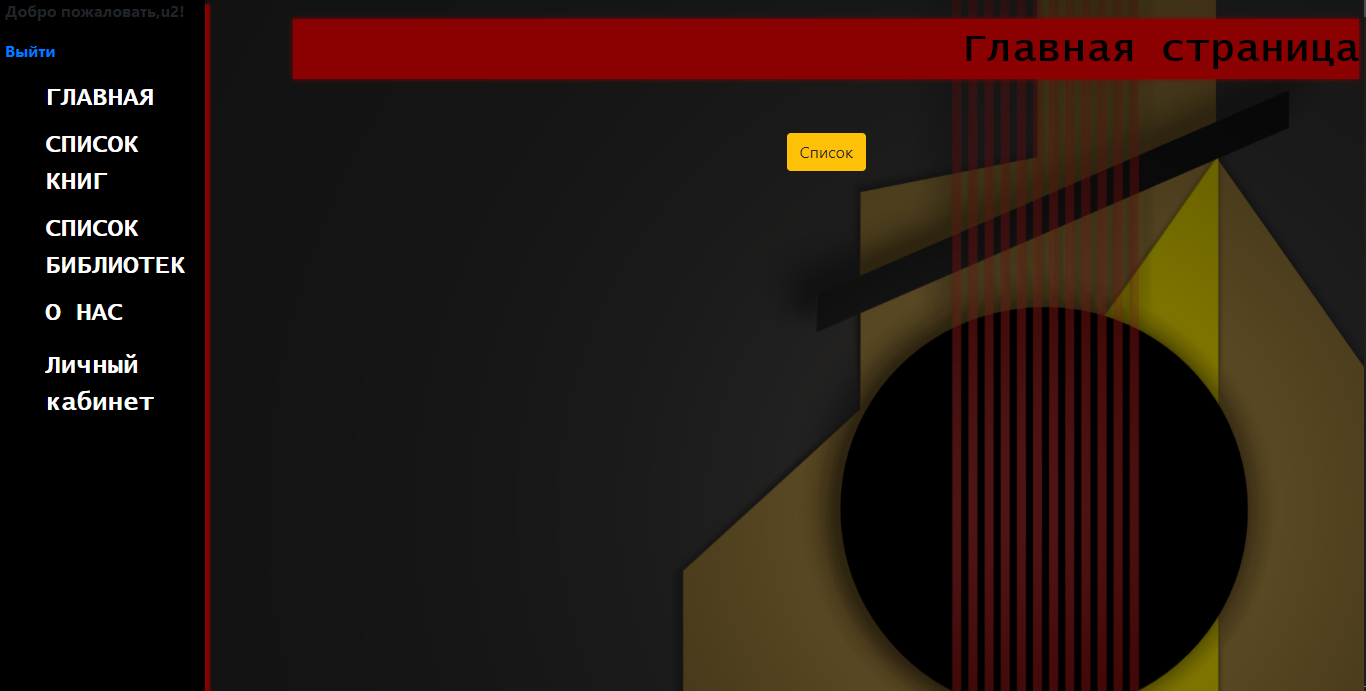


Рис.5.1 – Главная страница

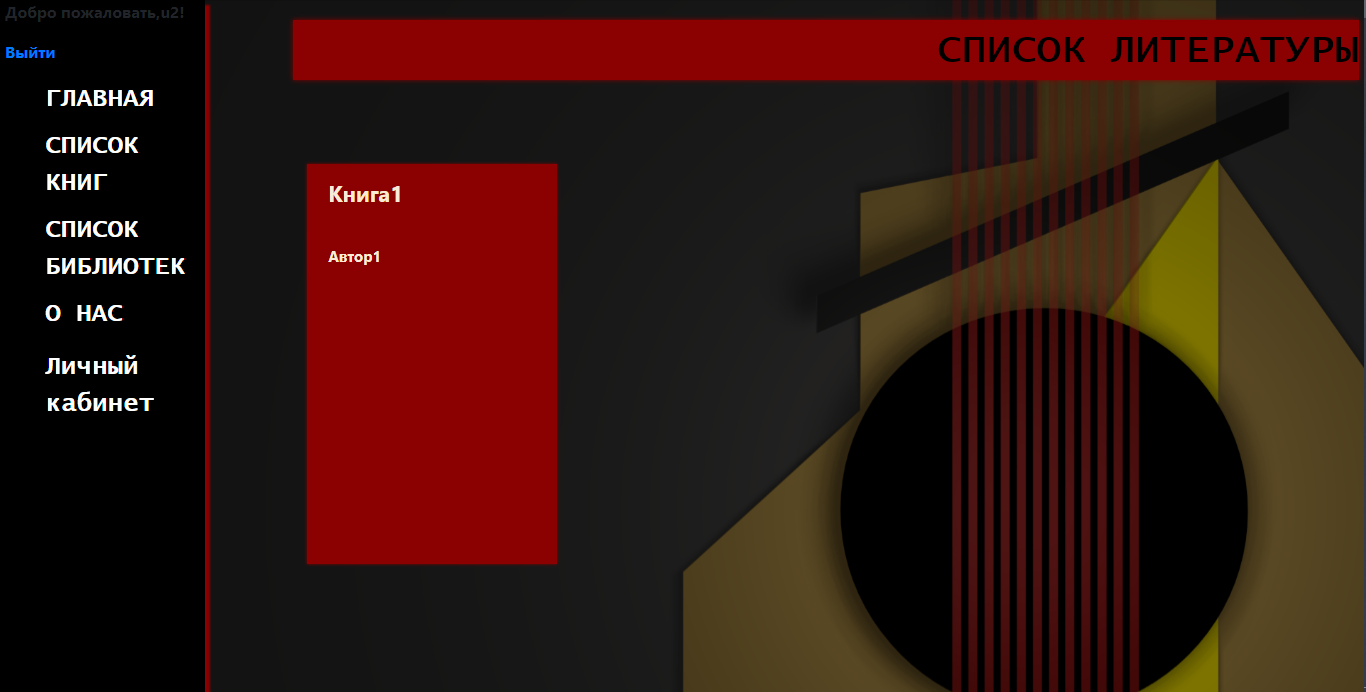


Рис.5.2 – Список библиотек

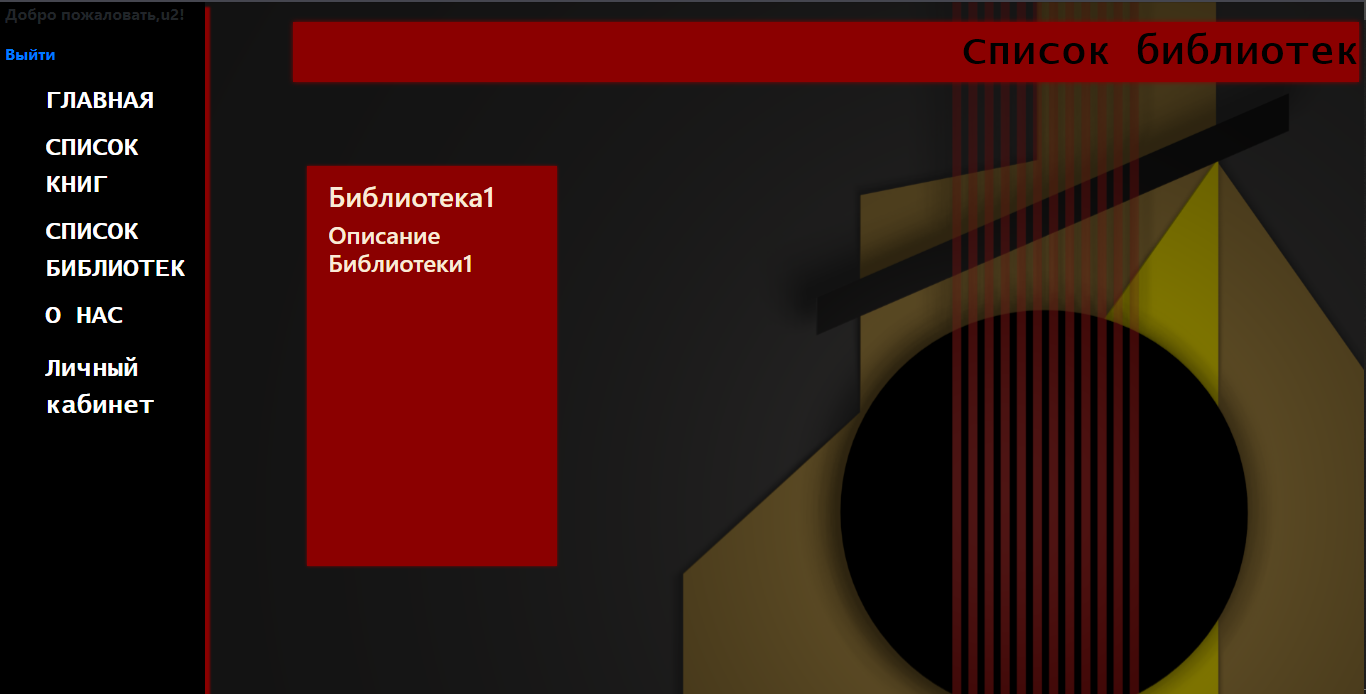


Рис.5.3 – Список библиотек

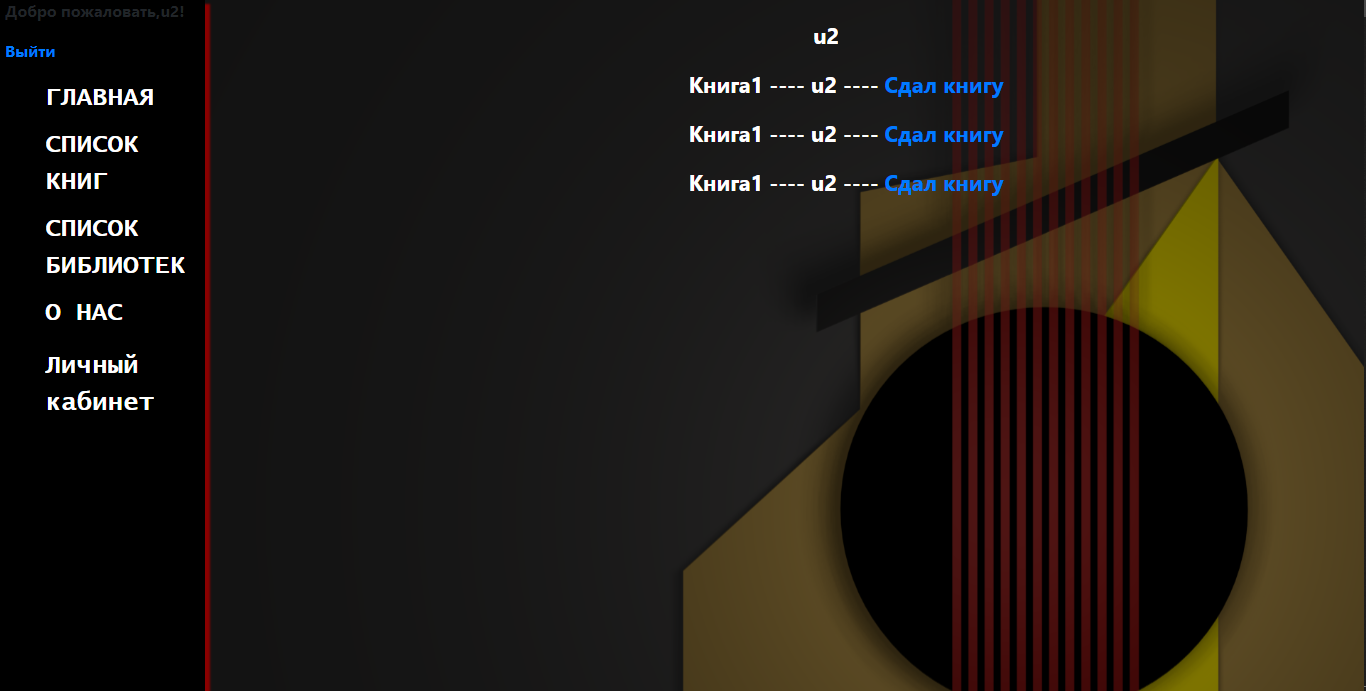


Рис.5.4 – Личный кабинет



Рис.5.5 – Подробное описание книги

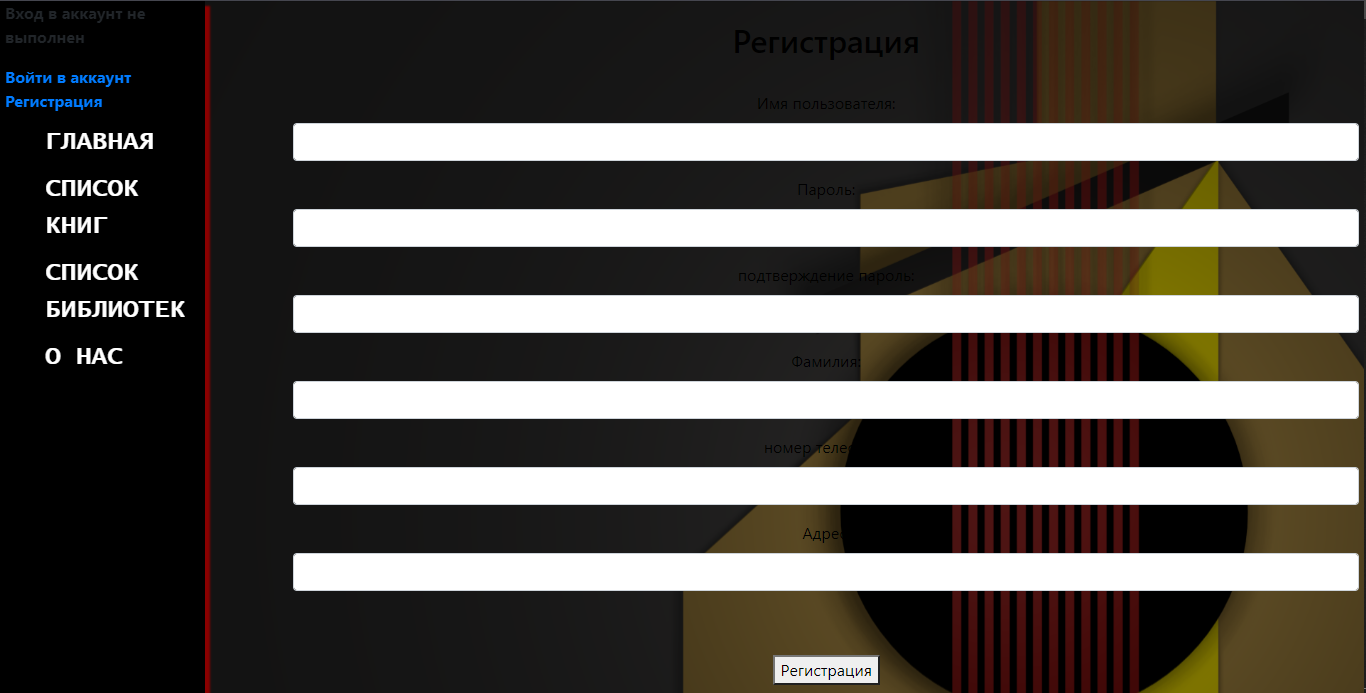


Рис.5.6 – Страница регистрации

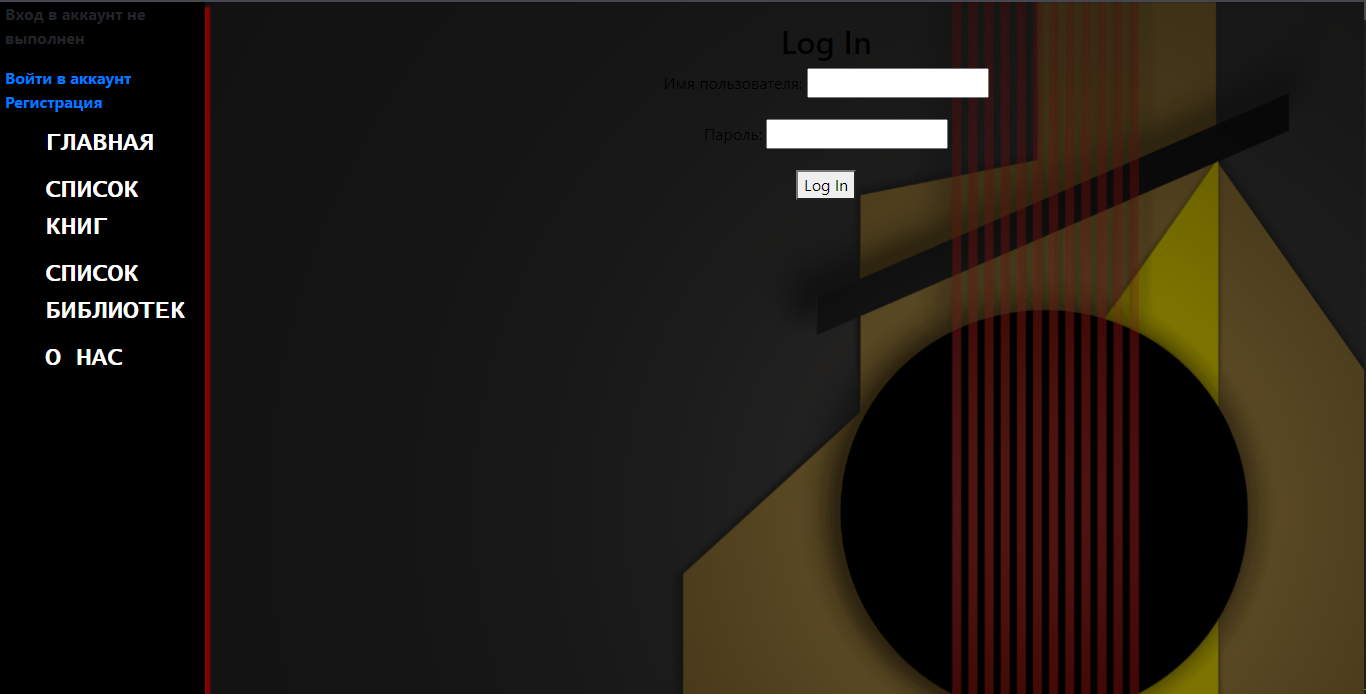


Рис.5.7 – Страница авторизации

В базе данных хранятся данные о пользователях (ФИО, номер телефона, адрес, электронная почта, прикрепленные книги); данные о библиотеках; данные о книгах (название, автор, описание, дата публикации).

### 5.1.Реализация регистрации и авторизации аккаунта

Код регистрации и авторизации аккаунта представлен в листинге кода 5.1.1, листинге кода 5.1.2, листинге кода 5.1.3, листинг кода 5.1.4.

Листинг кода 5.1.1 – файл forms.py

class SignupUser(UserCreationForm):

username = forms.CharField(label='Имя пользователя', widget=forms.TextInput(attrs={'class':'form-control'}))

password1 = forms.CharField(label='Пароль', widget=forms.PasswordInput(attrs={'class':'form-control'}))

password2 = forms.CharField(label='подтверждение пароль', widget=forms.PasswordInput(attrs={'class': 'form-control'}))

middle\_name = forms.CharField(label='Фамилия', widget=forms.TextInput(attrs={'class': 'form-control'}))

number = forms.CharField(label='номер телефона', widget=forms.TextInput(attrs={'class': 'form-control'}))

residence = forms.CharField(label='Адрес', widget=forms.TextInput(attrs={'class': 'form-control'}))

class Meta:

model = User

fields=('username','password1','password2', 'middle\_name','number','residence')

def save(self):

user = super(SignupUser, self).save()

customer = Customers(

user = user,

middle\_name = self.cleaned\_data['middle\_name'],

number=self.cleaned\_data['number'],

residence=self.cleaned\_data['residence'],

date = datetime.date(2012, 1, 1)

)

customer.save()

return user

class SignupCustomer(UserCreationForm):

middle\_name = forms.CharField(label='Фамилия', widget=forms.TextInput(attrs={'class':'form-control'}))

class Meta:

model = Customers

fields = ('middle\_name',)

number = forms.CharField(label='Номер телефона', widget=forms.TextInput(attrs={'class': 'form-control'}))

class Meta:

model = Customers

fields = ('number',)

residence = forms.CharField(label='Адрес', widget=forms.TextInput(attrs={'class': 'form-control'}))

class Meta:

model = Customers

fields = ('residence',)

Листинг кода 5.1.2 – Файл models.py

class Customers(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

date = models.DateField('Год рождения')

middle\_name = models.CharField(max\_length=150, blank=True)

number = models.CharField(max\_length=150, blank=True)

residence = models.CharField(max\_length=150, blank=True)

books = models.ManyToManyField(Articles)

def \_\_str\_\_(self):

return self.user.username

class Meta:

verbose\_name = 'Пользователь'

verbose\_name\_plural= 'Список Пользователей'

class librarian(models.Model):

user=models.ForeignKey(User, on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return self.title

class Meta:

verbose\_name = 'Библиотекарь'

verbose\_name\_plural= 'Список библиотекарей'

Листинг кода 5.1.3 – Файл url.py

urlpatterns = [

path('signup/', register, name='signup'),

path('Profile/', views.Profile, name='profile'),

]

Листинг кода 5.1.4 – Файл views.py

class SignUpView(generic.CreateView):

form\_class = UserCreationForm

success\_url = reverse\_lazy('home')

template\_name = 'signup.html'

def register(request):

if request.method=='POST':

user\_form = SignupUser(request.POST)

if user\_form.is\_valid():

user\_form.save()

return redirect('home')

else:

user\_form = SignupUser()

return render(request, 'accounts/signup.html',{'user\_form':user\_form})

def Profile(request):

customer = Customers.objects.get(user\_id=request.user.id)

return render(request, 'accounts/Profile.html', {

'customer': customer,

'customers': Customers.objects.all(),

'customer\_books': customer.books.all(),

'books': Articles.objects.all(),

'all\_books': CustomersBooks.objects.all()

})

### 5.2.Реализация закрепления книги

Прикрепление книги происходит после выбора книги пользователем на странице «Список книг». В подробном описании книги необходимо нажать кнопку «Взять книгу». После это к пользователю будет прикреплена выбранная им книга.

Код процесса прикрепления представлен в листинге кода 5.2.1, листинге кода 5.2.2.

Листинг кода 5.2.1 – Файл models.py

class CustomersBooks(models.Model):

customer = models.ForeignKey(to='accounts.Customers', on\_delete=models.CASCADE)

article = models.ForeignKey(Articles, on\_delete=models.CASCADE)

Листинг кода 5.2.2 – Файл views.py

def add\_book(request, pk):

customer = Customers.objects.get(user\_id=request.user.id)

l = CustomersBooks(customer=customer, article=Articles.objects.get(pk=pk))

l.save()

return redirect('home')

### 5.3.Реализация страницы «Список книг»

На данной странице выводится информация о книгах в виде плиток с кратким описанием. При наведении на какую либо плитку она увеличивается в размерах и изменяет цвет.

Код реализации страницы списка книг представлен на листинге кода 5.3.1.

Листинг кода 5.3.1 – Файл list\_index.html

{% extends 'main/layout.html' %}

{% block title %}

Список

{% endblock %}

{% block content %}

<div class="features">

<p class="namecat">СПИСОК ЛИТЕРАТУРЫ</p>

{% for el in list %}

<a {% if user.is\_authenticated %} href="{% url 'book\_detail' el.id %}"{% endif %} class="name\_book"><div class="alert alert-warning"><p class="zag">{{ el.title }}</p><p class="auctor">{{ el.anons }}</p></div></a>

{% endfor %}

</div>

{% endblock %}

## Тестирование

Тестирование проводится в браузере YandexBrowser. При разрешении экрана 1366х768.

Открытие сайта по ссылке <https://danilparunov.pythonanywhere.com/>. Результат открытия ссылки представлен на Рис.6.1.

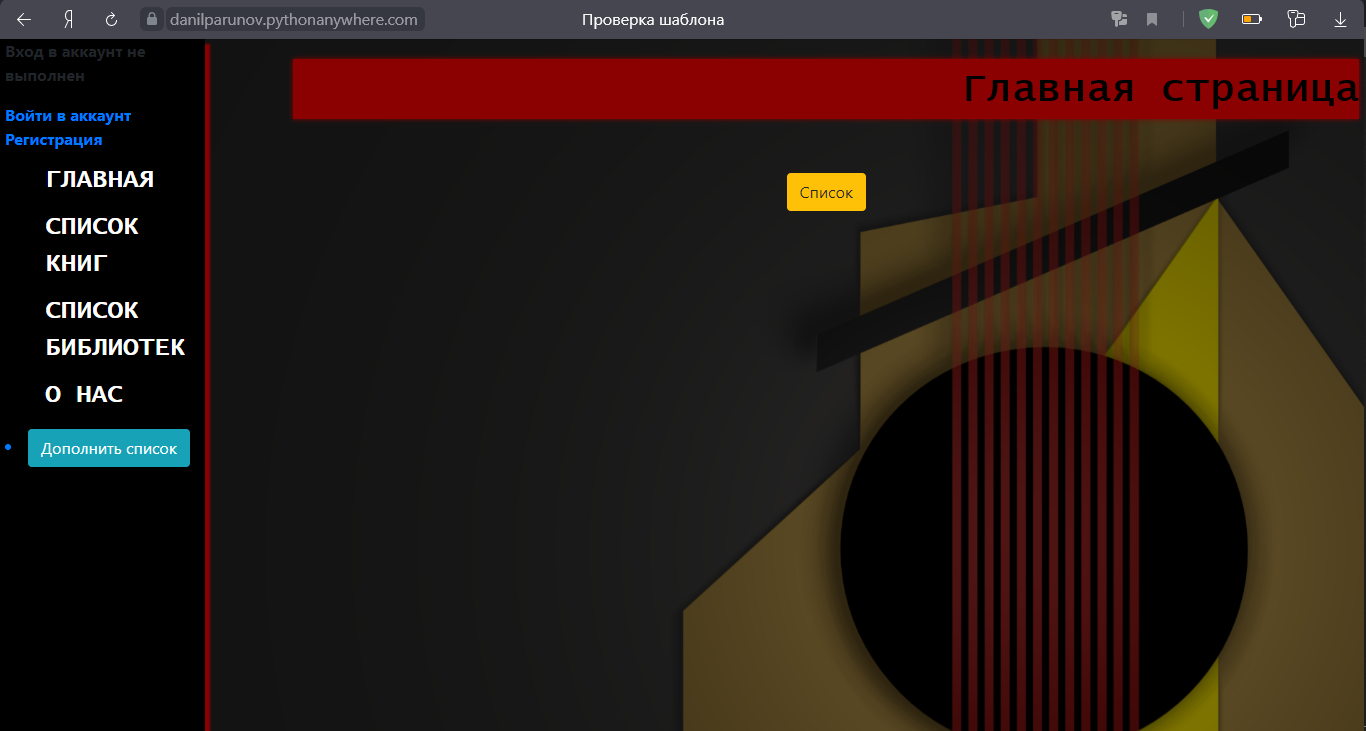


Рис.6.1 – Страница сайта

В панели разработчика отсутствуют ошибки. Панель разработчика представлена на Рис.6.2.

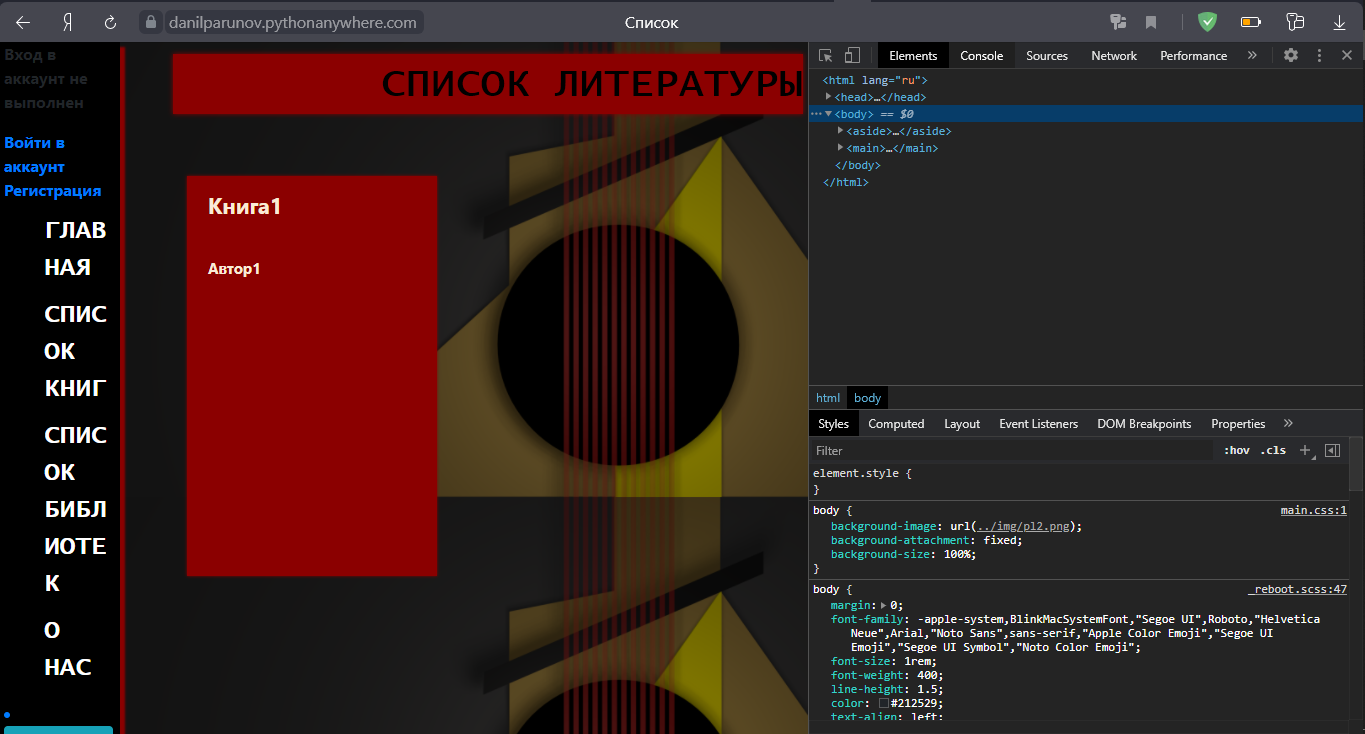


Рис.6.2 – Панель разработчика

Загрузка страницы происходит в пределах 2 секунд. Однако присутствует длительная отрисовка заднего фона по причине скорости загрузки изображения с сервера PythonAnywhere.

### 6.1.Тестирование регистрации нового пользователя и добавления его в базу данных.

Для этого перейдем по ссылке в правом верхнем углу под именем «Регистрация». На странице регистрации введем необходимые данные (Рис.6.1.1). После чего нажмем на кнопку регистрация. При успешной регистрации мы будем переадресованы на главную страницу сайта. При неправильном заполнений данных, будет выведен текст с описанием ошибки.

Для авторизации переходим по ссылке под именем «войти в аккаунт» в левом верхнем углу (Рис.6.1.2). При успешном входе в аккаунт, в левом верхнем углу будет присутствовать текст «Добро пожаловать, «Имя пользователя» » и появится пункт «Личный кабинет» в панели навигации (Рис.6.1.3). Код регистрации пользователя представлен в листинге кода 6.1.1, листинге кода 6.1.2, листинге кода 6.1.3.

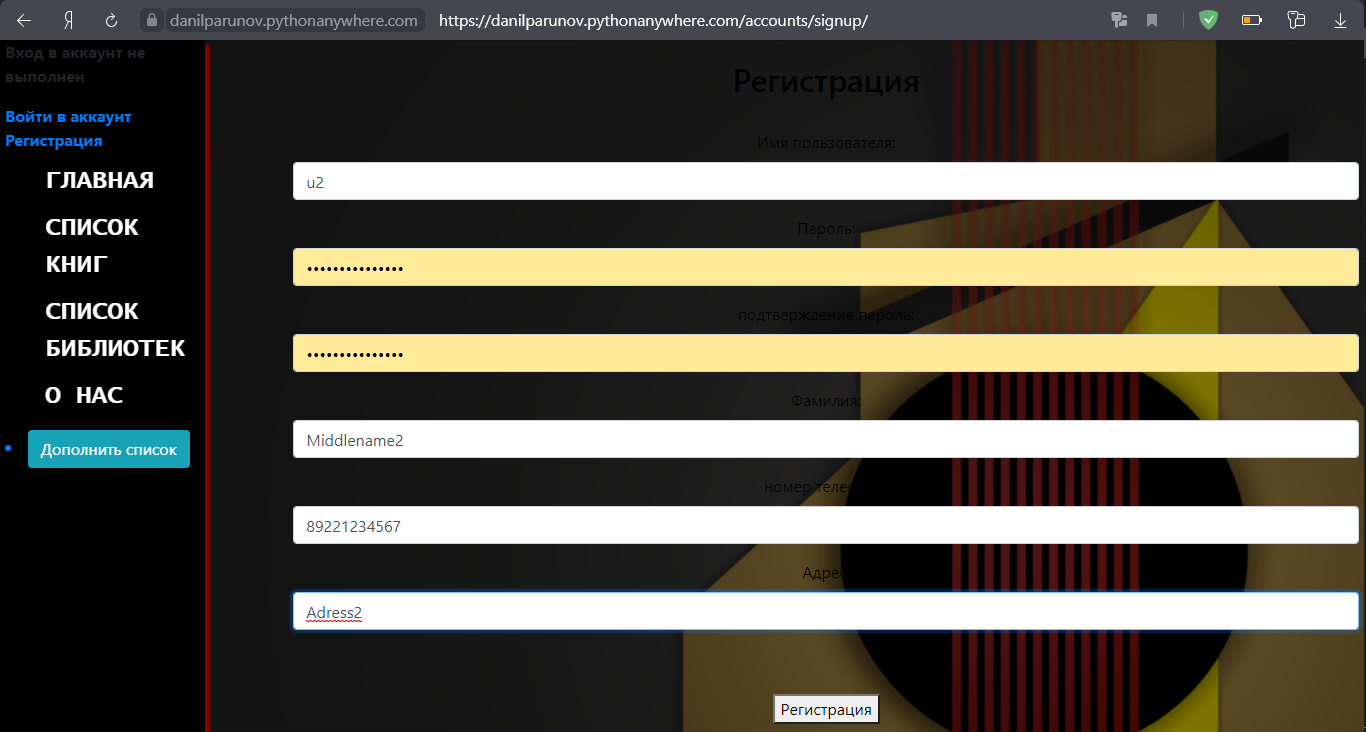


Рис. 6.1.3– Данные для регистрации

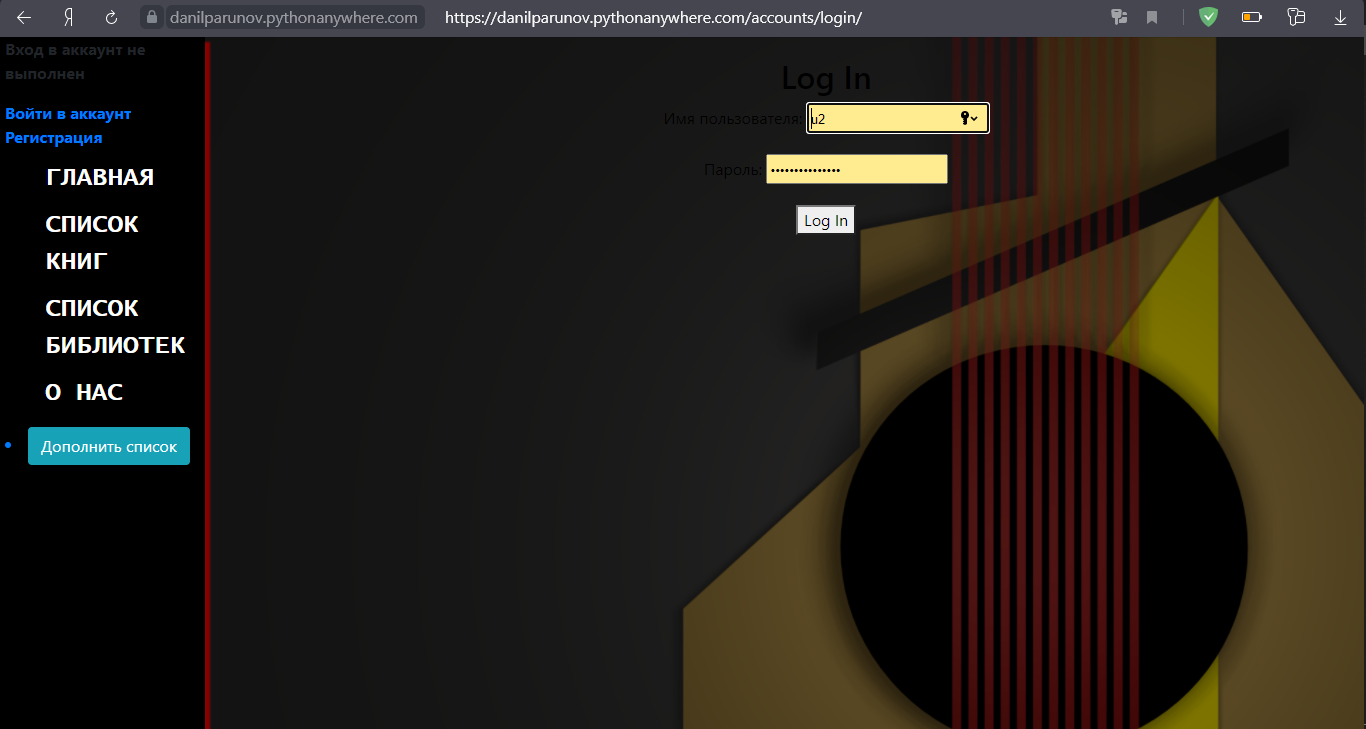


Рис.6.1.2 – Страница авторизации

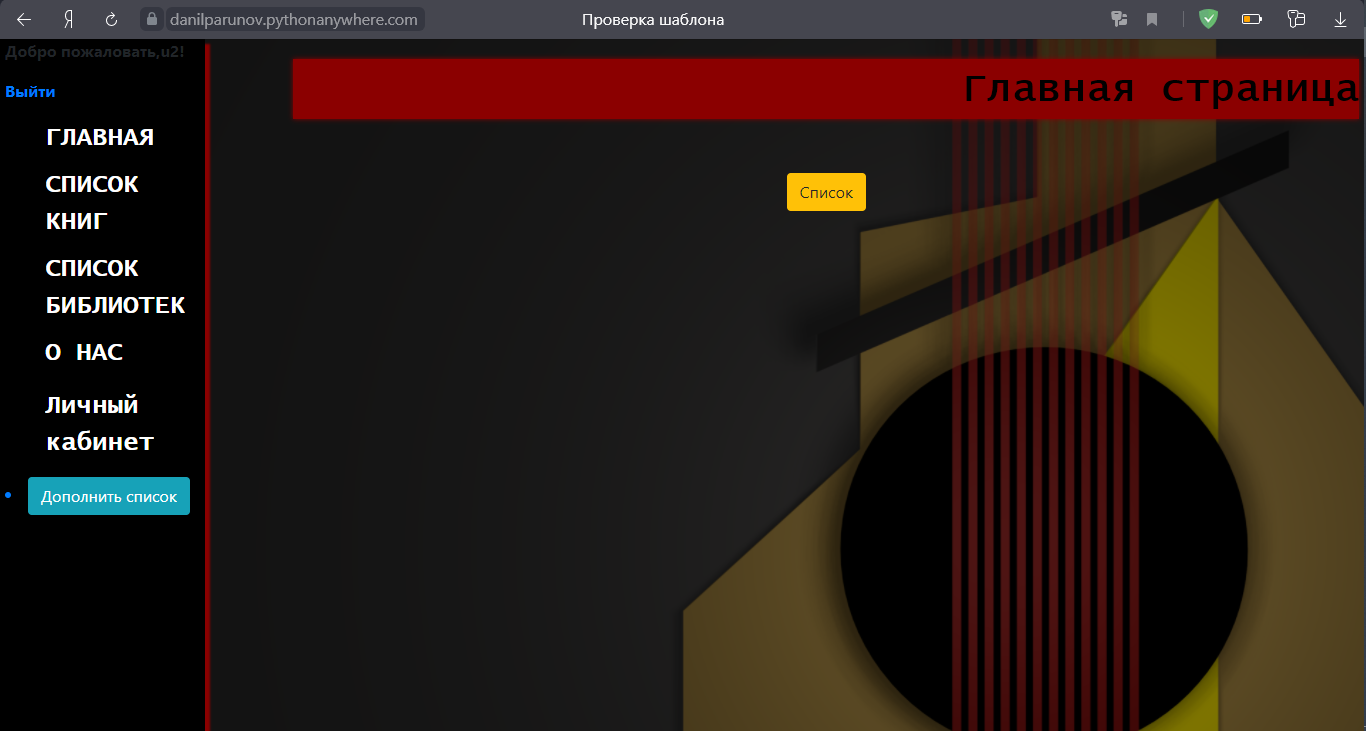


Рис.6.1.3 – Успешный вход в аккаунт

Листинг кода 6.1.1 – файл forms.py

class SignupUser(UserCreationForm):

username = forms.CharField(label='Имя пользователя', widget=forms.TextInput(attrs={'class':'form-control'}))

password1 = forms.CharField(label='Пароль', widget=forms.PasswordInput(attrs={'class':'form-control'}))

password2 = forms.CharField(label='подтверждение пароль', widget=forms.PasswordInput(attrs={'class': 'form-control'}))

middle\_name = forms.CharField(label='Фамилия', widget=forms.TextInput(attrs={'class': 'form-control'}))

number = forms.CharField(label='номер телефона', widget=forms.TextInput(attrs={'class': 'form-control'}))

residence = forms.CharField(label='Адрес', widget=forms.TextInput(attrs={'class': 'form-control'}))

class Meta:

model = User

fields=('username','password1','password2', 'middle\_name','number','residence')

def save(self):

user = super(SignupUser, self).save()

customer = Customers(

user = user,

middle\_name = self.cleaned\_data['middle\_name'],

number=self.cleaned\_data['number'],

residence=self.cleaned\_data['residence'],

date = datetime.date(2012, 1, 1)

)

customer.save()

return user

Листинг кода 6.1.2 – файл views.py

class SignUpView(generic.CreateView):

form\_class = UserCreationForm

success\_url = reverse\_lazy('home')

template\_name = 'signup.html'

def register(request):

if request.method=='POST':

user\_form = SignupUser(request.POST)

if user\_form.is\_valid():

user\_form.save()

return redirect('home')

else:

user\_form = SignupUser()

return render(request, 'accounts/signup.html',{'user\_form':user\_form})

Листинг кода 6.1.3 – файл models.py

class Customers(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

date = models.DateField('Год рождения')

middle\_name = models.CharField(max\_length=150, blank=True)

number = models.CharField(max\_length=150, blank=True)

residence = models.CharField(max\_length=150, blank=True)

books = models.ManyToManyField(Articles)

def \_\_str\_\_(self):

return self.user.username

class Meta:

verbose\_name = 'Пользователь'

verbose\_name\_plural= 'Список Пользователей'

### 6.2. Тестирование добавления книги в базу данных

Перейдем в пункт «Дополнить список» в панели навигации. Для этого необходимо быть авторизованным по учетной записью «LIB1», т.к. эта учетная запись имеет доступ к пункту «Дополнить список». После перехода откроется страница с формой для добавления книги (Рис.6.2.1). Введем данные показанные на Рис.6.2.2.

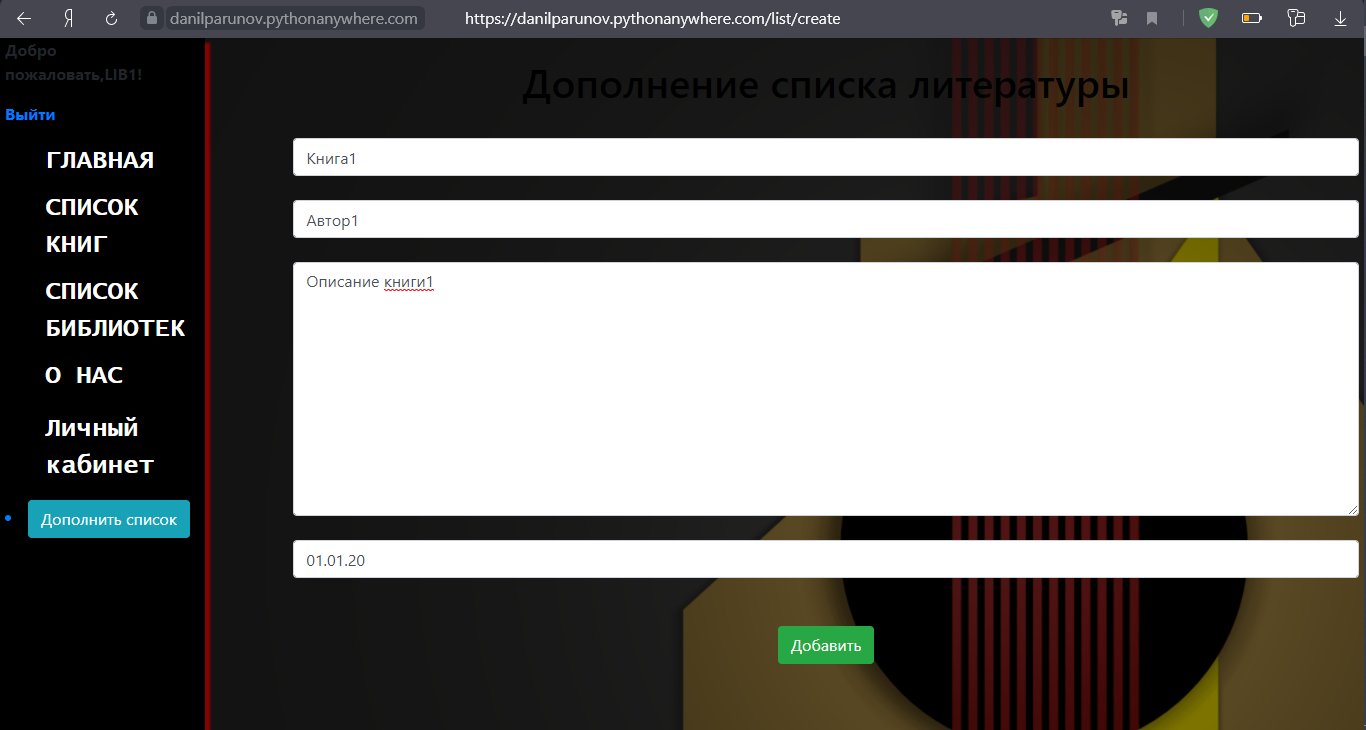


Рис.6.2.2 – Страница добавления книги

После перейдем на страницу списка книг и проверим наличие добавленной книги (Рис.6.2.3).

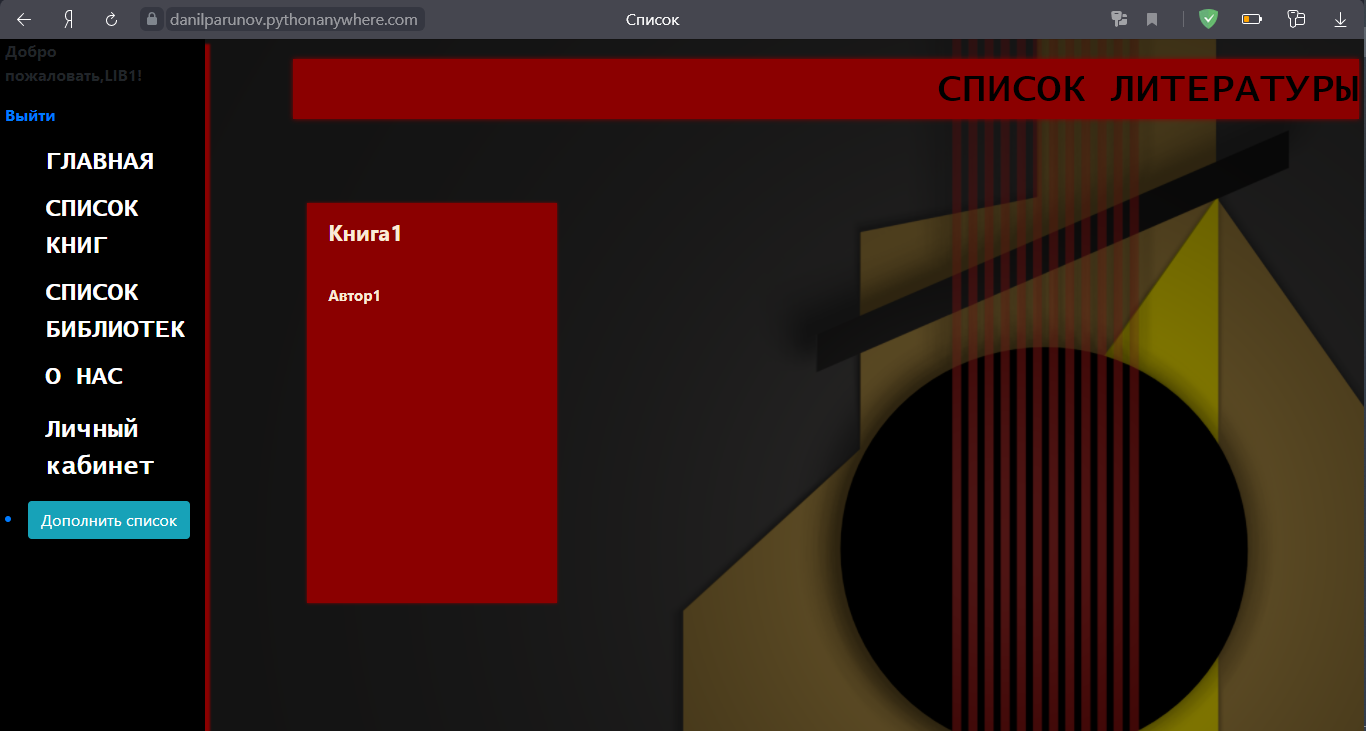


Рис.6.2.3 – Добавленная книга

### 6.3.Тестирование прикрепления книги к пользователю

Для выполнения данной операции необходимо зарегистрироваться или авторизоваться, в противном случае переход к подробному описанию книги и ее закреплению будет невозможен из-за ограничения. После переходим на страницу со списком книг. Выбираем нужную книгу и кликнем по ней, тем самым переходим на страницу подробной информации об этой книге (Рис.6.3.1).

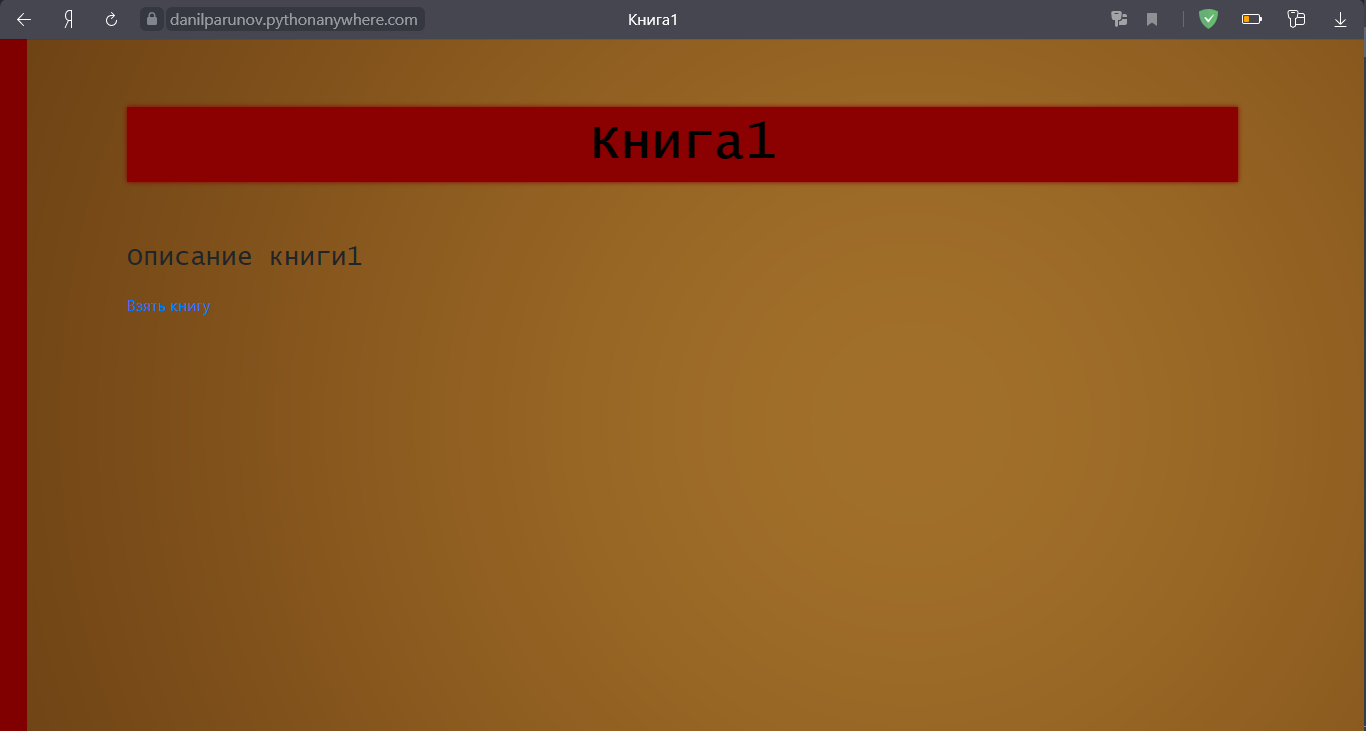


Рис.6.3.1 – Подробная информация о книге

Для прикрепления книги нажимаем на текст «Взять книгу». После мы будем перенаправлены на главную страницу, а в личном кабинете появится строка с названием книги Рис.6.3.2.

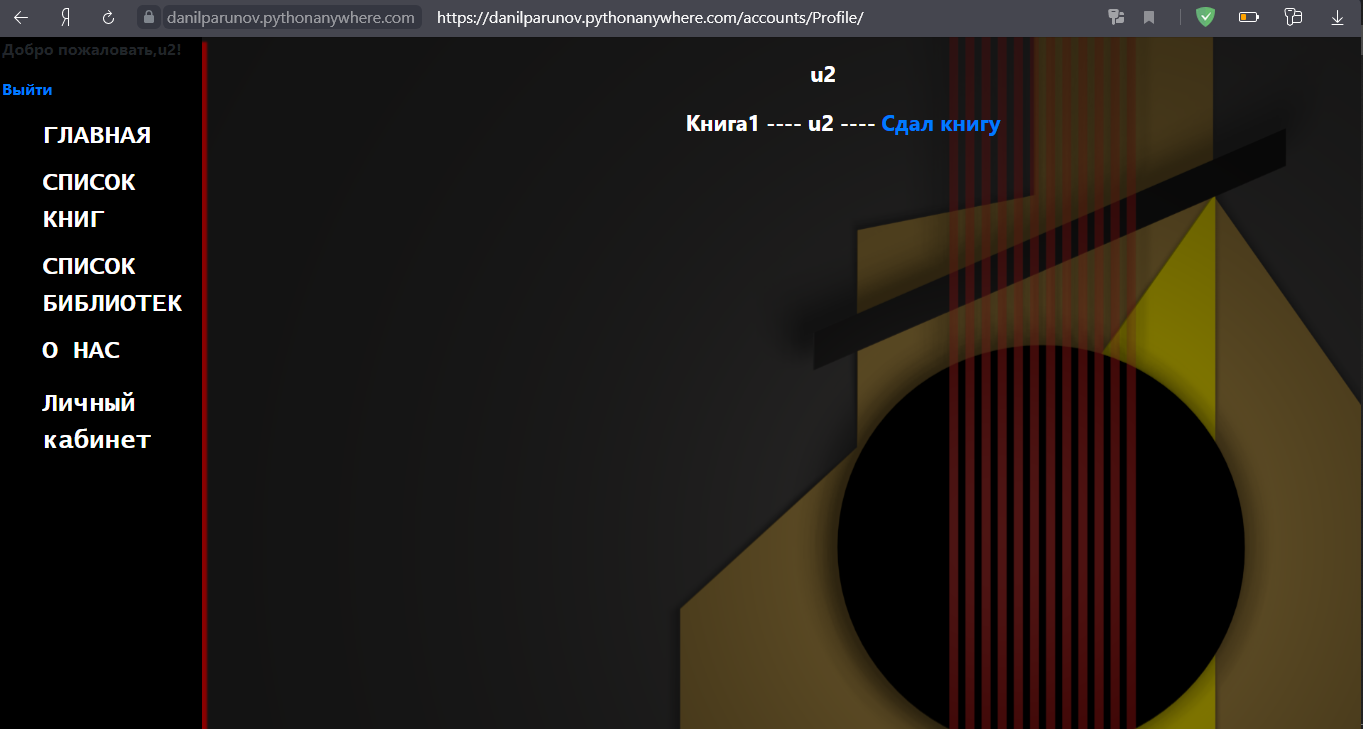


Рис.6.3.2 – Личный кабинет

# Заключение

В ходе разработки был создан сайта «Агрегатор библиотек города», при помощи которого можно найти и взять в личное пользование бумажную копию книги на определенный срок.

Были выполнены следующие задачи:

* Реализовать регистрацию и авторизацию учетной записи;
* Создать базу данных;
* Реализовать страницу списка книг;
* Реализовать личный кабинет пользователя;
* Реализовать закрепление книги;
* Реализовать страницу добавления книги.

Так же построены DFD диаграмма , ER модель, диаграмма прецедентов, диаграмма классов. Проведено тестирование основных функций сайта.

# Список использованной литературы

1. Технология разработки программного обеспечения: [Электронный ресурс]. URL: https://znanium.com/catalog/document?id=336552. (Дата обращения: 5.01.2021).
2. DFD методология. Нотация, принципы моделирования: [Электронный ресурс]. URL: https://www.nazametku.com/dlia-raboty/dfd-методология-нотация-принципы-модел/. (Дата обращения: 4.01.2021).
3. ER диаграмма (ERD):определение и обзор: [Электронный ресурс]. URL: https://www.lucidchart.com/pages/ru/erd-диаграмма. (Дата обращения: 7.01.2021).
4. Основы UML: [Электронный ресурс]. URL: https://pro-prof.com/archives/2594. (Дата обращения: 7.01.2021).
5. UML-диаграммы классов: [Электронный ресурс]. URL: https://prog-cpp.ru/uml-classes/. (Дата обращения: 9.01.2021).
6. Python documentation: [Электронный ресурс]. URL: https://www.python.org/doc/. (Дата обращения: 10.01.2021).
7. Django documentation: [Электронный ресурс]. URL: https://docs.djangoproject.com/en/3.1/. (Дата обращения: 4.01.2021).