УНИВЕРСИТЕТ ИТМО

International Culture & Technology Studies
online multimedia journal

ITMO UNIVERSITY

Культура и технологии
электронный мультимедийный журнал

Journal Homepage: http://cat.ifmo.ru
ISSN 2587-800X

Адрес статьи / To link this article: http://cat.itmo.ru/ru/2023/v8-i1/392

База данных по истории и технологии бумаги: состав, структура, особенности

А. В. Корчилава, Е. И. Носова

Санкт-Петербургский институт истории РАН, Россия

kor4ilava@itmo.ru, katerinanossova@gmail.com

Аннотация. В данной статье описывается проектирование и создание базы данных по истории и технологии бумаги. В связи быстрым развитием большого количества различных методик исследования бумаги базы данных становятся особенно необходимы, так как редко один специалист владеет всеми методами: чаще образец проходит серию исследований у разных специалистов, разными методами. В результате информация рассредоточивается, и необходимо свести ее воедино, чтобы получить обобщенные сведения об образце для последующего анализа и исследования. Ключевой особенностью этой базы данных является ее гибкость и адаптивность, так как технологии изучения и исследования бумаги развиваются, а вместе с ними и увеличиваются объемы информации, необходимые для хранения одного объекта. Для полноценного анализа каждый объект проходит множество исторических, реставрационных и естественнонаучных исследований и, чтобы вся информация хранилась надежно и структурированно, создается база данных на языке SQL с применением системы управления базами данных PostgreSQL. База содержит в себе историческую справку, фотографии и расчеты, информацию о предыдущей реставрации, снимки и файлы спектров, результаты количественного анализа и многие другие характеристики. Для комфортного пользования всеми специалистами, база размещается на сайте с помощью фреймворка Django, работающем на языке Python.

Ключевые слова: бумага, водяной знак, база данных, СУБД, PostgreSQL, Django, PgAdmin4, модели, миграции, представления (views), HTML шаблон, Jinja шаблонизатор

1. Введение

Бумага на протяжении многих лет являлась самым распространенным материалом для письма. Без ее изобретения и широкого внедрения не были бы возможны распространение книгопечатания и, как следствие, связанная с этим информационная революция, перевернувшая сознание человечества. Этот факт делает бумагу не просто носителем текста, но важнейшей составной частью развития письменной и печатной культуры. От ее свойств, технологических особенностей, требований к условиям производства, его объемов зависело развитие центров книгопечатания и образования, цивилизации в целом. В производстве бумаги было занято значительное количество людей, оно было стимулом для технологического развития, так как требовало механизации для увеличения выхода продукта. Все это делает изучение технологии бумаги самоценной и самодостаточной тематикой исследования.

Роль бумаги в построении цивилизации и сохранении памяти и необходимость ее изучения были давно осознаны учеными различных специализаций [1], [2]. Первоначально исследование бумаги связывалось с задачами датировки рукописей по филиграням. Не менее важным было изучение бумаги в процессе предреставрационных исследований и работ, направленных на выработку мер по стабилизации и превентивной консервации документов на бумажных носителях. В настоящий момент все больше прикладное значение исследований сменяется пониманием необходимости изучения бумаги как многогранного комплекса, в котором пересекаются самые разные задачи и представители разных дисциплин. Бумага становится точкой пересечения материаловедения, производства, торговли, книжной культуры и моды. В результате такой подход порождает большое количество информации, которое необходимо хранить, структурировать, обрабатывать, делать доступным максимально широкому научному сообществу. Задача создания базы данных по бумаге ставилась несколько ранее [3], но в настоящий момент в открытом доступе существуют лишь базы данных по филиграням [2]. Разумеется, филигрань — важнейший атрибут бумаги, несущий в себе значительное количество информации. Однако она ни в коем случае не исчерпывает всей полноты материальности бумаги. Таким образом, на сегодняшний день желательно создание базы данных, которая рассматривала бы не отдельные аспекты бумаги как материала, а позволяла бы сконцентрировать максимальное количество информации с целью ее последующей обработки и анализа.

2. Методы и программное обеспечение

Основное назначение современных баз данных — удобное, структурированное и понятное хранение больших объемов информации на компьютере [4].

Работа с базами данных на компьютерах осуществляется с помощью специальных систем управления базами данных (СУБД), которые связывают сами данные с приложением или программным обеспечением (рис. 1).

Рис. 1. Схема работы СУБД

СУБД выполняет две основные функции:

- предоставляет доступ к базе данных для оператора или программного обеспечения;
- через СУБД происходят управление и обработка данных базы.

В качестве СУБД в данном проекте была выбрана свободная PostgreSQL, версия 15.0 и программный продукт PgAdmin 4 для администрирования и разработки базы данных. PostgreSQL — объектно-реляционная система управления базами данных, которая работает на языке SQL, поддерживает широкие возможности и обладает высокой производительностью [5]. Некоторые из множества преимуществ PostgreSQL, которые послужили толчком для выбора именно этой СУБД это:

- в первую очередь, открытый исходный код PostgreSQL. Т. е. СУБД находится в открытом доступе, любой пользователь может скачать и начать работу с базами данных;
- PostgresSQL работает с различными форматами данных, как обычными числовыми и текстовыми, так и с форматами файлов, картинок, JSON файлами и даже геометрическими данными координат геопозиций;
- PostgreSQL — надежная СУБД и соответствует требованиями ACID (атомарность от англ. *atomicity*, согласованность от англ. *consistency*, изолированность от англ. *isolation*, устойчивость от англ. *durability*). В такой СУБД данные хранятся надежно и не теряются в процессе выполнения запросов в реальном времени;

- в PostgreSQL возможно работать с данными больших размеров, а размер базы данных и вовсе не ограничен;
- легкая расширяемость и управление PostgreSQL.

Обычные пользователи обращаются к БД не через СУБД, а через приложение, например, сайт. Основной задачей в данном проекте было создание базы данных, удобной для использования любым пользователям, как физикам, химикам, так и историкам, реставраторам, хранителям. Для этого требуется создать приложение, которое будет предоставлять доступ к базе данных и содержать весь необходимый функционал.

Для разработки веб-приложения и сайта базы данных лаборатории был выбран фреймворк Django, который работает на языке Python. Версия Django в данном проекте 3.2.7, а также использовалось большое количество различных дополнительных библиотек и модулей python и Django, такие как numpy, matplotlib, psycopg2 и многие другие [6]. Можно выделить несколько причин выбора именно Django среди остальных фреймворков:

- наличие открытого исходного кода, т. е. все находится в открытом пользовании и пользование фреймворком доступно каждому;
- наличие бесплатной поддержки и большой, доступной документации;
- удобство в использовании;
- безопасность создаваемого приложения;
- умеренная гибкость кода, т. е. Django позволяет разработчику создавать индивидуальные гибкие проекты, но в то же время не даст совершить ошибку на корневом уровне (например, чтобы база данных работала только на определенных операционных системах);
- кроссплатформенный фреймворк;
- система «всё включено». В Django реализуются как frontend (интерфейс и взаимодействие с пользователем), так и backend (внутренняя часть продукта на сервере) разработка, благодаря чему появляется возможность создавать и управлять сайтом и базой данных в одном месте, что значительно упрощает работу разработчика.

Основная задача фреймворка — соединить в себе подключение к базе данных, модели и маршруты сайта, сам сайт. Для этого структура фреймворка включает в себя различные компоненты, такие как URL-маршрутизаторы, которые передают запрос пользователя с сайта в представления фреймворка (views), сами представления, которые обрабатывают запрос и обращаются с ним к базе данных, модель (ORM — Object Relational Mapping), предоставляющую нужную информацию и передающую ее в представления и HTML-шаблоны, которые визуализируют и предоставляют пользователю запрошенную информацию. Примерная схема работы фреймворка представлена на рис. 2 [6].

Подробнее все этапы работы с приложением и базой данных описаны в следующей части.

3. Основная часть

3.1. Проектирование базы данных

На первом этапе происходит проектирование базы данных, т. е. определение необходимых отношений (таблиц), характеристик объектов, типов полей и связей между ними [7]. В приложении 1 представлена таблица с данными, характеристиками объектов, которые будут храниться в базе данных лаборатории и тип поля, который требуется для ввода информации. Этот список полей разработан на основании оборудования и возможностей, имеющихся в настоящий момент в распоряжении Лаборатории комплексного исследования рукописных памятников Санкт-Петербургского института истории РАН, на базе которой осуществляется данная работа. Был учтен опыт Международной ассоциации историков бумаги [3].

Для дальнейшей работы с фреймворком (рис. 2) надо понимать, в каком формате и какие типы полей присваивать тем или иным характеристикам в PostgreSQL. Типы полей отличаются в зависимости от выбранной СУБД, но именно в PostgreSQL есть возможность работать с большим количеством данных разного формата. Так, для текста и чисел присваиваются поля varchar(n), text и integer (smallint, bigint и т. д.), а для хранения даты стандартное поле date. Картинки, файлы и

фотографии хранятся в Django, а в самой базе данных хранится строка — ссылка на нужную картинку. Поля, которые представляют собой выбор из словаря, хранятся как числовой формат, т. е. пользователь выбирает определенный пункт словаря — индекс. Также для создания поля по введению количественного анализа использовано поле типа jsonb, характерное именно PostgreSQL. Jsonb хранит данные JSON (JavaScript Object Notation) в бинарном формате [6].

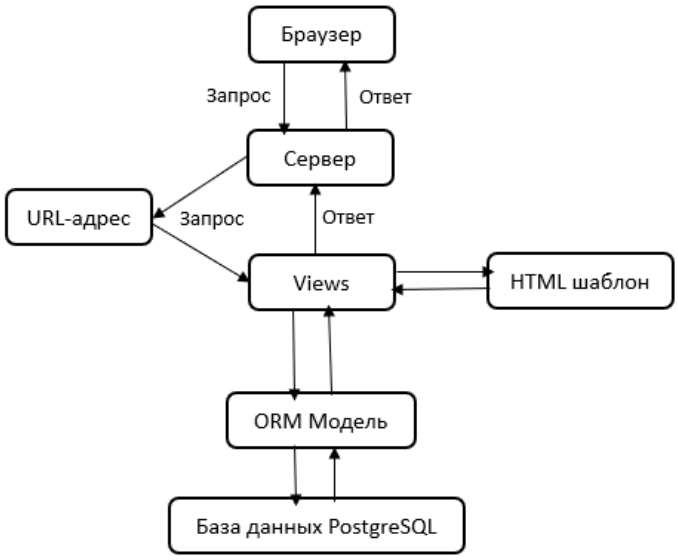

Рис. 2. Схема формирования запросов пользователя и ответов во фреймворке

3.2. Создание проекта и приложения на Django

Чтобы не допустить несостыковок и неполадок в версиях Python, Django и других пакетов, разработка происходит в виртуальном окружении (venv). Благодаря виртуальному окружению можно изолировать проект от существующих данных на компьютере.

Разработка проекта происходит на сервере и на данный момент используется локальный сервер компьютера разработчика — localhost 127.0.0.1. Для запуска сервера используется команда «python manage.py runserver».

На первом этапе выполняется команда «django-admin startproject DataBase», после чего создается проект DataBase, который хранит в себе все необходимые файлы. Одним из наиболее важных файлов является settings.py, где устанавливаются основные глобальные настройки проекта. Здесь задается секретный ключ для выгрузки на общедоступный сервер, вывод ошибок, ограничение доменов для доступа к сайту или базе данных, регистрируются приложения, указывается конкретная база данных, с которой будет идти работа в проекте, и устанавливается язык проекта [10].

Для работы с разными категориями сайта в Django создаются приложения, которые содержат файлы работы с базой данных, HTML шаблоны страниц сайта и файлы для переадресации по сайту. Приложение создается с помощью команды «python manage.py startapp “название приложения”», и проект добавляется автоматически. Для связи приложения с сайтом, требуется зарегистрировать его в файле settings.py, INSTALLED_APPS.

На данный момент в проекте находится два приложения — main, которое отвечает за сайт, главную страницу, контакты, информацию, а также news, которое отвечает за работу с базой данных и ее отображение на сайте.

Для того чтобы приложение отображалось на веб-сайте и работало по нужной ссылке, в файле urls.py добавляется новая ссылка:

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls')),
    path('news/', include('news.urls')),
] + static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

Теперь приложение будет подключаться при переходе по ссылке, а также Django будет ссылаться на необходимые HTML шаблоны для отображения страницы сайта.

3.3. Создание модели базы данных

Для работы с самой базой данных было создано новое приложение — news с определенным URL и отличными от приложения main HTML шаблонами. Для создания новых таблиц в базе данных (моделей) в Django используется файл models.py, где прописываются различные классы — таблицы [11]. В каждом классе прописываются поля таблицы, их типы полей, подписи и многие другие характеристики. В базе данных лаборатории на данный момент находится одна таблица — Objects. Пример формирования класса моделей и первые строчки Object представлены ниже:

```
class Objects(models.Model):
      cipher_section = models.PositiveSmallIntegerField("Шифр
     секции",    choices=SECTION, blank=True, null=True)
      cipher_fond = models.PositiveSmallIntegerField("Шифр
     коллекция/фонд", choices=FOND, blank=True, null=True)
      cipher_fond_number = models.BigIntegerField("Номер
     коллекции/фонда", blank=True, null=True)
      cipher_case_number = models.BigIntegerField("Номер
     дела", blank=True, null=True)
      cipher_list_number = models.CharField("Номер листа",
     max_length=250, blank=True)
      sample_number = models.CharField("№ образца(для образцов
     без шифра)", max_length=250, blank=True)
      year = models.PositiveSmallIntegerField("Год",
     blank=True, null=True)
      month = models.PositiveSmallIntegerField("Месяц",
     choices=MONTH, blank=True, null=True)
      day = models.PositiveSmallIntegerField("День",
     choices=DAY, blank=True,  null=True)
      author = models.CharField("Автор", max_length=250,…
```

После названия поля и ссылки на models указывается тип поля (PositiveSmallIntegerField, BigIntegerField, CharField, ImageField и т. д.), название поля в базе данных, параметры, необходимые под различные типы полей. Например, в поле CharField указывается максимальная длина возможного введенного текста max_length=250, в поле, где требуется выбор из словаря, указывается ссылка на словарь choices=MONTH, а сам словарь также прописывается в файле models.py.

Особый интерес представляют поля для ввода количественного анализа вещества, т. е. процентного соотношения веществ в образце. Поскольку заранее неизвестно, сколько какого

вещества будет в том или ином образце, поле будет динамическим — меняться от образца к образцу. Формат ячейки для ввода JSONB — тип поля, характерный именно для PostgreSQL и по сути являющийся json, только более компактный и удобный в использовании. Поле ввода в панели администрирования представляет собой обычное поле ввода json файлов формата ключ-значение: {key: value, key: value, key: value}. Объекты JSON также могут быть вложены друг в друга и представлять собой более сложную ступенчатую структуру, но для нашего случая с количественным значением достаточно будет одномерного файла.

Для того чтобы пользователю, который добавляет объект, было удобнее вводить данные количественного анализа, поле ввода JSON файла на сайте будет выглядеть не как текстовое поле, а как пара полей — ключ: значение — и возможность изменять количество этих пар (рис. 3 и 4).

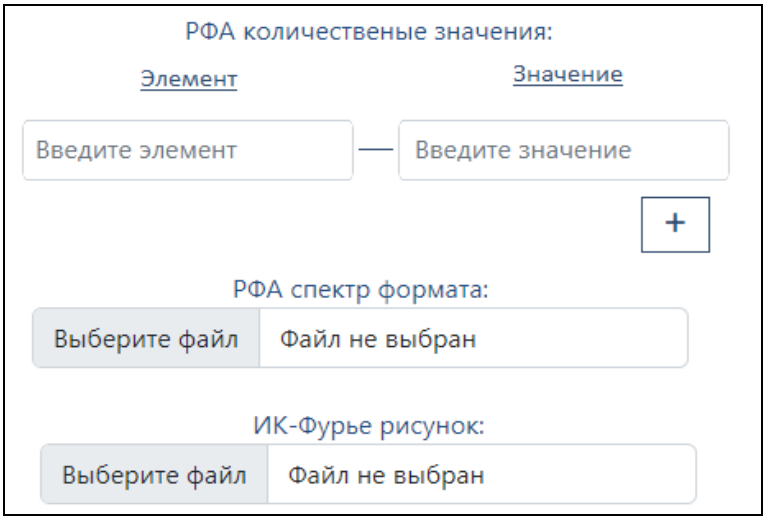

Рис. 3. Пример поля заполнения количественного анализа

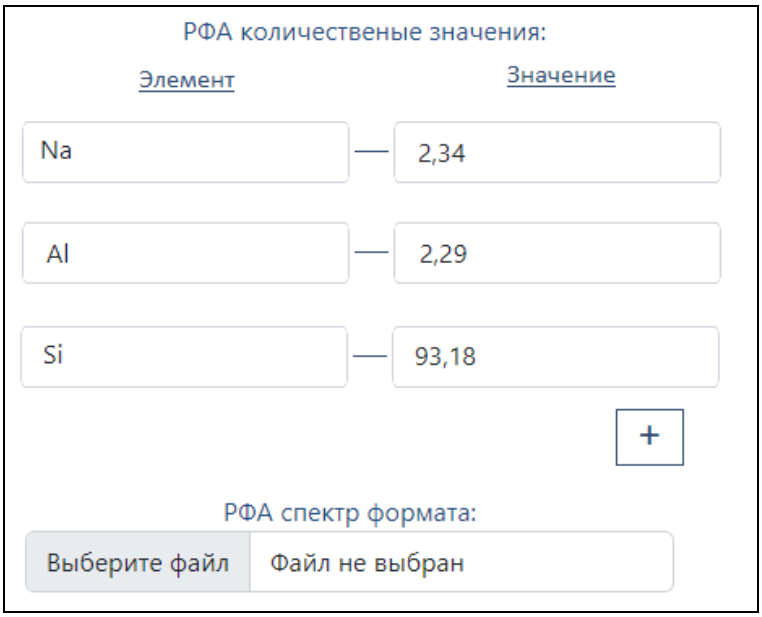

Рис. 4. Пример ввода нескольких веществ для количественного анализа

Для того чтобы была возможность прикреплять несколько фотографий (рис. 5), создаются дополнительные модели с внешним ключом Foreign Key (связь многие к одному — Many To One) для связи с главной моделью Objects [11]. Класс ForeignKey принимает два обязательных параметра:

– ссылка на класс модели, с которой будет происходить связь;

– функция on_delete, которая накладывает ограничения при удалении записи из базы данных.

Например:

```
class Filpic(models.Model):
    object = models.ForeignKey(Objects,
on_delete=models.CASCADE, null=True, related_name='imgs')
    image = models.ImageField(upload_to='try_fil_image/')
```

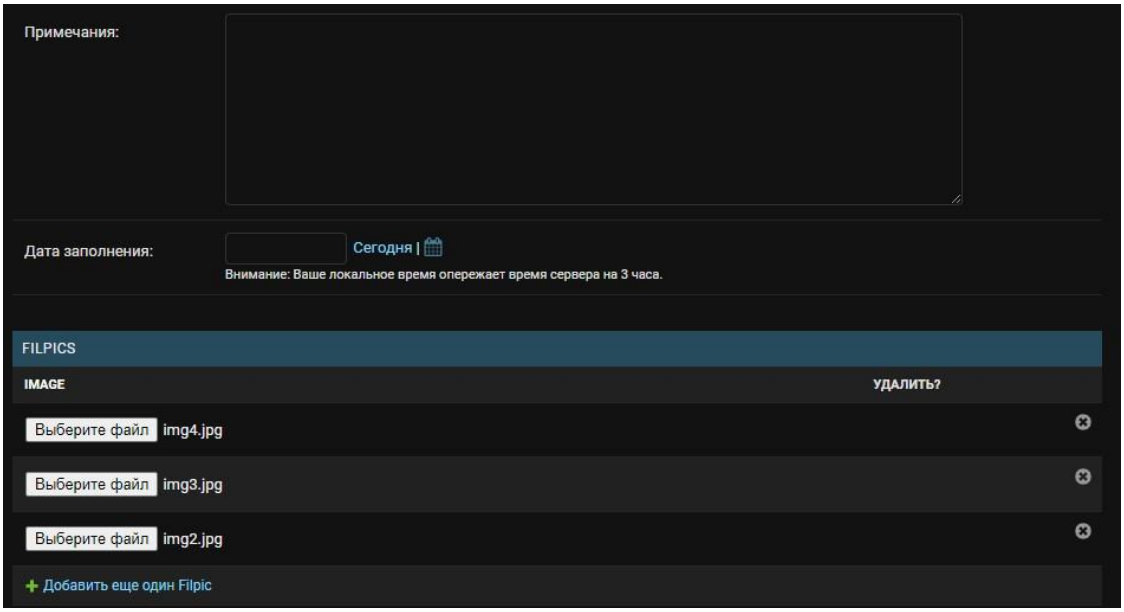

Рис. 5. Пример оформления прикрепления множества фотографий

Подсчет данных через формулу по введенным параметрам осуществляется с помощью функций, прописанных вручную. Например, функция подсчета площади листа:

```
def square(self):
    if self.size_width is not None and
self.size_height is not None:
        a = self.size_width
        b = self.size_height
        c = (a * b) / 1000000
        return round(c, 4)
    else:
        self.square = None
        return print('Размеры объекта не указаны')
```

С помощью метода __repr__ возвращается ID номер каждого объекта:

```
def __repr__(self):
    return self.id
```

После того как все поля в models.py прописаны, таблица в базе данных еще не создана. Для формирования таблицы в PostgresSQL необходимо выполнить процедуры миграции — перенос данных в СУБД и синхронизация проекта с базой данных. Для начала создаются файлы миграций, для этого в терминале проекта выполняется команда: python manage.py makemigrations. После появляется файл миграции в приложении, который описывает таблицу в базе данных. Затем миграции проводятся с помощью команды python manage.py migrate.

3.4. Работа с панелью администратора Django и PostgreSQL

Для перехода в панель администрирования Django необходимо перейти по URL-адресу сервера с добавочным окончанием /admin. Для администрирования сайта необходимо зарегистрироваться, т. е. создать пользователя, у которого будут права администратора. Для этого в терминале выполняется команда python manage.py createsuperuser и вводятся данные пользователя (имя, пароль и т. д.) В панели администрирования Django находятся все те таблицы, с которыми возможно работать в контексте базы данных, а также управление пользователями, зарегистрированными в базе.

Чтобы работать с таблицей Objects, созданной в базе данных, необходимо зарегистрировать ее в панели администратора admin.py:

```
admin.site.register(Objects, ObjectsAdmin)
```

Панель администрирования Django выглядит следующим образом (рис. 6, 7 и 8):

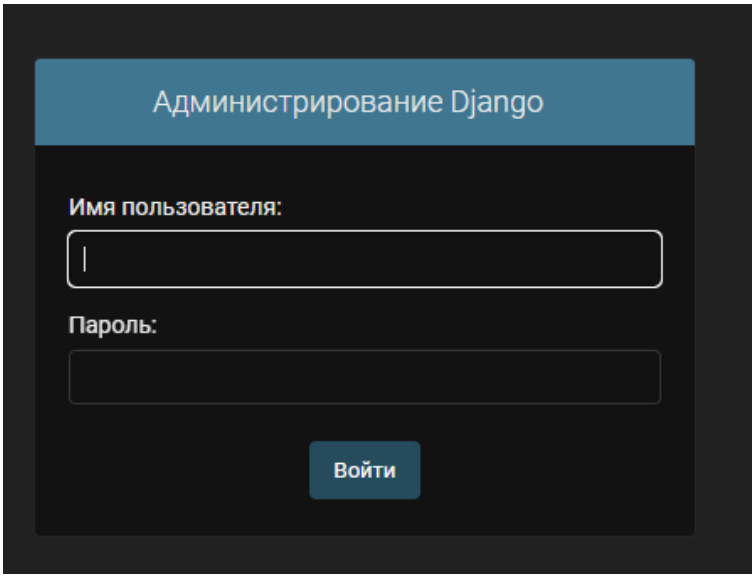

Рис. 6. Вход в панель администрирования Django

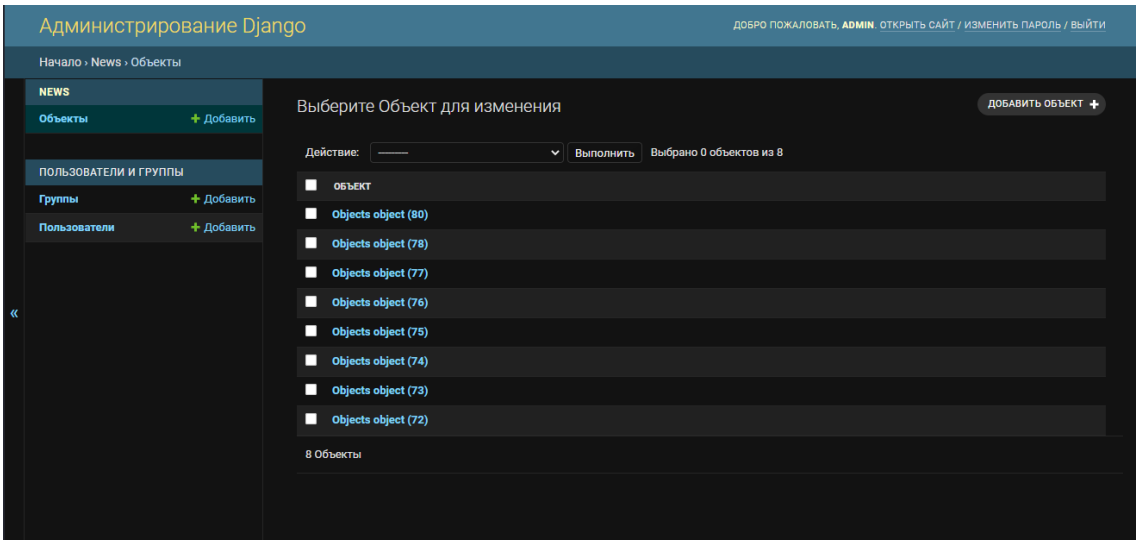

Рис. 7. Панель администрирования Django. Список доступных объектов базы данных.

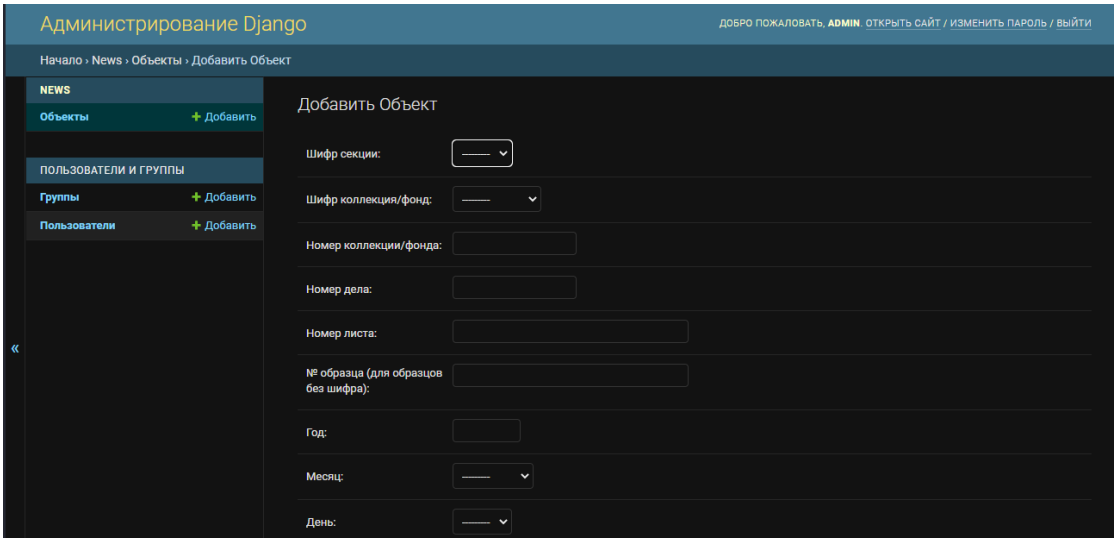

Рис. 8. Панель администрирования данных. Добавление нового объекта

По сути, работать с базой данных, добавлять, редактировать и удалять объекты можно уже на этом этапе, но для более удобного общего пользования базой, требуется вывести все это на сайт, а также создать удобный интерфейс.

Одновременно с выполнениями миграций, модель Objects формируется и в PostgreSQL. Для работы и просмотра базы данных в ее первоначальном виде используется приложение PgAdmin4 [12]. С помощью запросов на языке SQL (рис. 9) можно точно также администрировать, управлять и изменять базу данных через это приложение [13].

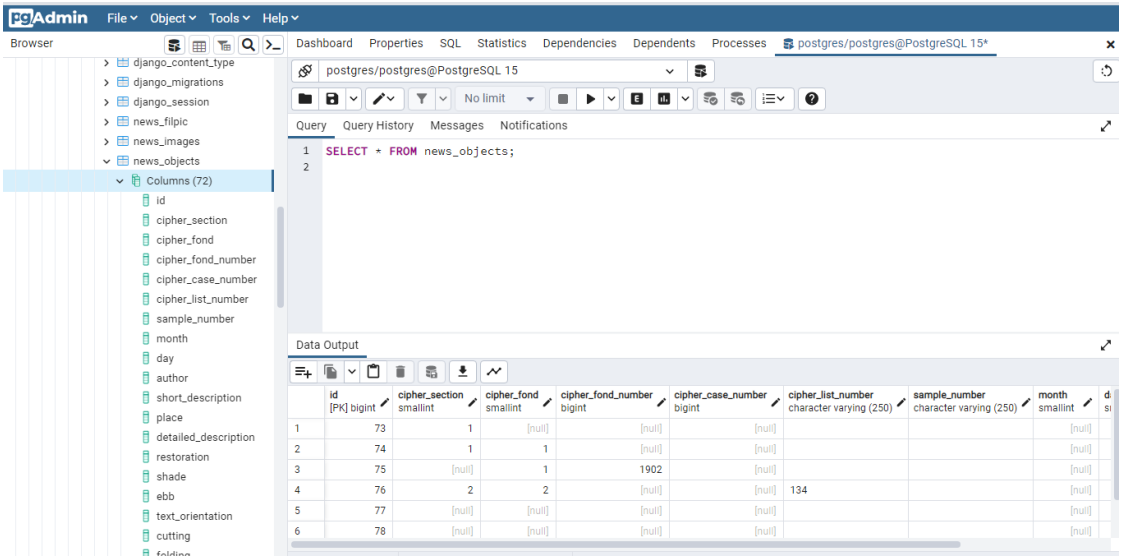

Рис. 9. Таксономия визуальных руководств Управление базой данных PostgreSQL через приложение PgAdmin4 и SQL запросы.

3.5. Вывод записей из базы данных на сайт

Для вывода записей необходимо импортировать модель, которую мы прописали выше в файле views.py приложения:

```
from .models import Objects
```

А для вывода всех объектов, сортированных по дате публикации, необходимо передать в нужный шаблон записи из базы данных [14]:

```
def news_home(request):
    news = Objects.objects.order_by('data')
    return render(request, 'news/news_home.html',
{'news': news})
```

Для красивого и понятного отображения моделей, в файле models.py также прописывается метод __str__, который возвращает заголовок объекта:

```
def __str__(self):
    return self.id
```

А также необходимо прописать вывод каждого объекты в шаблоне HTML с помощью шаблонизатора и стили CSS для корректного отображения [15]:

```
<div class="features">
    <h1>Объекты</h1>
    {% if news %}
        {% for el in news %}
            <div class="alert alert-primary">
                <h3>№{{ el.id }}</h3>
                <p>{{ el.intro }}</p>
                <a href="{% url 'object-detail' el.id %}"
class="btn btn-secondary">Смотреть детали</a>
            </div>
        {% endfor %}
    {% else %}
        <p>Нет доступа к объектам</p>
    {% endif %}
</div>
```

3.6. Форма добавления объекта на сайте

Для связи модели, таблицы и сайта необходимо прописать специальную форму, в которой указывается, как будет отображаться и передаваться запрос пользователя с сайта. Для этого создается файл forms.py, куда импортируются модели и формы. В файле создается класс ModelForm, в котором указывается модель, поля, которые будут отображаться на сайте, а также виджеты и характеристики каждого отдельного поля:

```
class ObjectsForm(ModelForm):
    class Meta:
        model = Objects
        fields = ['cipher_section', 'cipher_fond',…,'notes',
'data']
        widgets = {
            "cipher_section": Select(attrs={'class': 'form-
select', 'placeholder': 'Шифр секции'}),
            "cipher_fond_number": TextInput(attrs={'class':
'form-control', 'placeholder': 'Номер коллекции/фонда'}),
            …
```

После указания конкретного поля в виджетах (поля соответствуют полям модели Objects) указывается класс, на основе которого формируется поле ввода и различные атрибуты этих классов в формате словаря.

В HTML-шаблоне подключается csrf token, который обеспечивает безопасную передачу данных из формы и указывается сама форма, а точнее, все ее поля, прописанные в forms.py:

```
<h1> Форма по добавлению объекта </h1>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <span>{{ error }}</span>
    <button class="btn btn-info"
type="submit">Добавить объект</button>
</form>
```

Добавляется также новое отслеживание URL адреса для страницы по созданию объекта, при переходе на который, вызывается метод create из файла views:

```
def create(request):
    error = ''
    if request.method == 'POST':
        form = ObjectsForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            print(form.cleaned_data)
            return redirect('/')
        else:
            error = 'Форма заполнена неверно'
    form = ObjectsForm()
    data = {
        'form': form,
        'error': error
    }
    return render(request, 'news/create.html', data)
```

Здесь проверяется метод POST (тот же, что указан в HTML шаблоне), проверяется правильно ли заполнена форма методом .is_valid(), а также происходит сохранение объекта и переадресация на указанную страницу [14]. Теперь во вкладке Объекты (рис. 10) сайта базы данных, хранится список всех добавленных объектов:

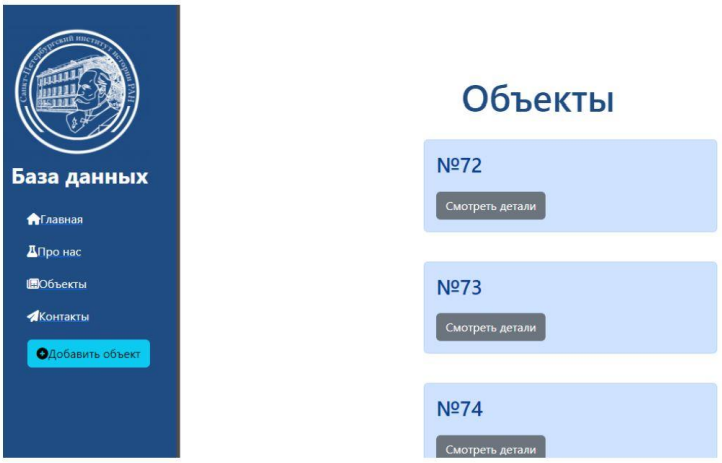

Рис. 10. Вкладка Объекта сайта базы данных

Для просмотра деталей каждого объекта необходимо сделать переадресацию на новую страницу с первичным ключом (ID) объекта (динамический параметр) при нажатии кнопки «Смотреть детали», а также прописать саму страницу с деталями. В данном проекте это сделано с помощью встроенного Django класса DetailView:

```
class ObjectsDetailView(DetailView):
    model = Objects
    template_name = 'news/details_view.html'
    context_object_name = 'object'
```

Здесь передаются сама модель, шаблон, в котором будет происходить просмотр деталей и контекстное имя, т. е. ключ, по которому будет передаваться запись.

Для кнопки «Смотреть детали» передаем параметр ID каждого объекта в URL адрес:

```
<a href="{% url 'object-detail' el.id %}"
class="btn btn-secondary">Смотреть детали</a>
```

В HTML шаблоне просмотра деталей передаются все характеристики объекта с помощью контекстного имени, указанного в классе DetailView (object):

```
<div class="features">
    <label for="id">ID</label>
    <h1>{{ object.id }}</h1>
    <label for="cipher_section">Шифр секции</label>
    <p>{{ object.get_cipher_section_display }}</p>
    <label for="cipher_fond">Шифр коллекция/фонд</label>
    <p>{{ object.get_cipher_fond_display }}</p>
    …
```

Для создания страниц редактирования объектов используется класс UpdateView, я также тот же HTML шаблон, что и при создании объекта. Для удаления объектов применялся класс DeleteView.

Для отслеживания URL адресов необходимо указывать динамические параметры в файле urls.py нашего приложения, т. к. количество и значения ID объектов постоянно меняются. В нашем случает это целое число <int:pk> и первичный ключ.

```
urlpatterns = [
    path('', views.news_home, name='news_home'),
    path('create', views.create, name='create'),
    path('<int:pk>', views.ObjectsDetailView.as_view(),
name='object-detail'),
    path('<int:pk>/update',
views.ObjectsUpdateView.as_view(), name='object-update'),
] + static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

4. Заключение

На данный момент база данных функционирует и уже пополняется различными тестовыми объектами. Каждый объект, на данный момент, содержит до 75 различных параметров и характеристик, в зависимости от проведенных исследований — историческая справка, предыдущая реставрация, результаты естественнонаучных исследование и многое другое. Параллельно с заполнением базы идут работы над сайтом, верстка и дизайн, а также улучшение и

добавление различных полей для каждого объекта (например, карта, для отметки места создания или нахождения).

Хочется отметить, что огромным преимуществом этой база данных является ее динамичность — при наличии прав администратора, можно добавлять и изменять поля для заполнения каждого объекта по мере появления новых технических возможностей лаборатории, а также новых технологии исследования бумаги.

Таким образом, созданная база данных рассматривает не только отдельные справки о бумаге, как о материале, а хранит в себе полноценную информацию для обработки, анализа и исследования.

Приложение 1

Таблица 1. Поля для базы данных и форматы их ввода

№	Название поля	Формат ввода
1.	ID	Автоматическая нумерация при вводе
2.	Шифр секция	Словарь
3.	Шифр коллекция/фонд	Словарь
4.	Шифр номер коллекции/фонда	Число
5.	Шифр_№ дела	Число
6.	Шифр_№ листа	Число
7.	№ образца (для образцов без шифра)	Буква + число
8.	Год	Диапазон
9.	Месяц	Словарь
10.	День	Словарь
11.	Автор	Текст
12.	Описание краткое	Текст
13.	Место создания	Словарь
14.	Место создания карта	Привязка к карте по координатам
15.	Подробное описание	Гиперссылка на основную бд
16.	Предыдущая реставрация	Словарь
17.	Оттенок	Словарь
18.	Отлив	Словарь
19.	Ориентация текста относительно верже	Словарь
20.	Обрезка	Словарь
21.	Сложение	Словарь
22.	Размер листа ширина, мм	Число
23.	Размер листа высота, мм	Число
24.	Площадь листа, м кв.	Подсчет через формулу
25.	Толщина листа средняя, мм	Число
26.	Толщина листа медиана, мм	Число
27.	Вес листа, гр.	Число
28.	Масса 1 кв.м., гр.	Подсчет через формулу
29.	Плотность, гр/см куб.	Подсчет через формулу
30.	Филигрань рис	Рисунок
31.	Филигрань высота, мм	Число
32.	Филигрань_описание (словесное)	Текст
33.	Филигрань_соответствие	Текст + число
34.	Филигрань_Бернштейн [8]	Гиперссылка
35.	Сохранность_филиграни	Словарь
36.	Контрмарка_рис	Рисунок
37.	Контрмарка_высота, мм	Число
38.	Контрамарка_описание	Текст
39.	Контрмарка_соответствие	Текст + число
40.	Контрмарка_Бернштейн [8]	Гиперссылка
41.	Сохранность_контрмарки	Словарь
42.	Расстояние м/у верже, мм	Число

№	Название поля	Формат ввода
43.	Расстояние м/у понтюзо, мм	Число
44.	Толщина верже, мм	Число
45.	Толщина понтюзо, мм	Число
46.	Дата отливочной формы формы	Число
47.	Белизна_средняя, cie	Число
48.	Яркость_средняя, %	Число
49.	Флуоресценция, %	Число
50.	Белизна_параметры измерений	Текст
51.	pH_среднее	Число
52.	pH_параметры измерений	Текст
53.	Состав по волокну Херцберг_рис	Рисунок
54.	Состав по волокну Графф С_рис	Рисунок
55.	Состав по волокну_описание	Словарь, мультивыбор
56.	Поляризационная микроскопия_в поляризованном свете	Рисунок
57.	Поляризационная микроскопия_в скрещенных николях	Рисунок
58.	Поляризационная микроскопия_параметры измерений	Текст
59.	РФА_рис спектра	Рисунок
60.	РФА_количественые значения	Символ + число
61.	РФА_спектр в формате	Файл
62.	РФА_параметры измерений	Текст
63.	ИК-Фурье_рис	Рисунок
64.	ИК-Фурье_спектр в формате	Файл
65.	ИК-Фурье_описание	Текст
66.	ИК-Фурье спектроскопия_параметры измерений	Текст
67.	СЭМ вторичные электроны_рис	Рисунок
68.	СЭМ обратно рассеянные_рис	Рисунок
69.	СЭМ_параметры измерений	Текст
70.	ЭДС_спектр рис	Рисунок
71.	ЭДС_спектр в формате	Файл
72.	ЭДС_количественные значения	Символ + число
73.	ЭДС_параметры измерений	Текст
74.	Литература	Текст
75.	Примечания	Текст

Литература

[1] Богданов А.П. Основы филиграноведения. История, теория, практика. М., 1999. 326 с.

[2] Балаченкова А. П., Цыпкин Д. О. Возможности технологического анализа исторических бумаг в источниковедческом исследовании памятников // Вестник Санкт-Петербургского университета. История. 2017. Т. 62. Вып. 2. С. 375–399.

[3] Есипова В.А. К вопросу о создании электронных баз данных филиграней // Археографический ежегодник. 2000. С. 70-93.

[4] Карпова И.П. Базы данных. Курс лекций и материалы для практических занятий. Учебное пособие. М.: Питер, 2013. 240 с

[5] Сушков А., Классен Н. СУБД PostgreSQL: почему её стоит выбрать для работы с данными и как установить // Блог Яндекс Практикума. URL: https://practicum.yandex.ru/blog/chto-takoe-subd-postgresql/#id1 (дата обращения: 02.03.2023).

[6] Django // SkillFactory.Блог [электронный ресурс]. 2022. URL: https://blog.skillfactory.ru/glossary/django/ (дата обращения 12.10.2022)

[7] Конюхов В.Г., База данных. Понятие, значение и роль в современном мире // Системные технологии. 2017. №24. С. 61-61.

[8] Bernstein. Memory of Paper. URL: https://www.memoryofpaper.eu/BernsteinPortal/appl_start.disp (дата обращения: 24.03.2023).

[9] PostgreSQL: The World's Most Advanced Open Source Relational Database // PostgreSQL (электронный ресурс). 2022. URL: https://www.postgresql.org/ (дата обращения 10.10.2022)

[10] Форсье Дж., Биссекс П., Чан У. Django. Разработка веб-приложений на Python. Пер. с англ. СПб.: Символ-Плюс, 2009. 456 с.

[11] Django ORM Cookbook // DJANGO.Fun [электронный ресурс]. 2019. URL: https://django.fun/en/docs/django-orm-cookbook/2.0/ (дата обращения: 25.10.2022)
[12] pgAdmin 4 Documentation. Release 6.20 (2023). URL: https://www.pgadmin.org/docs/pgadmin4/6.21/index.html (дата обращения 03.03.2023)
[13] Новиков Б.А., Горшкова Е.А., Графеева Н.Г. Основы технологий баз данных. Учебное пособие. 2е изд. М.: ДМК Пресс, 2020. 582 с.
[14] Русаков М. Django View – базовые представления // Помощник Python [электронный ресурс]. 2022. URL: https://pythonpip.ru/django/django-view-bazovye-predstavleniya (дата обращения: 15.10.2022)
[15] Get started with Bootstrap // GitHub [электронный ресурс]. 2022. URL: https://github.com/twbs/bootstrap/blob/v5.3.0-alpha1/site/content/docs/5.3/getting-started/introduction.md (дата обращения 15.12.2022)

Database on the History and Technology of Paper: Contents, Structure, Specific Features

A. V. Korchilava, E. I. Nosova

St Petersburg Institute of History, Russian Academy of Sciences, Russia

Abstract. This article describes the design and creation of a database on the history and technology of paper. Due to the rapid development of many different methods of paper research, databases are becoming especially necessary, since rarely one specialist owns all the methods: more often a sample undergoes a series of studies with different specialists, using different methods. As a result, the information is dispersed, and it is necessary to bring it together to obtain generalized information about the sample for subsequent analysis and research. The key feature of this database is its flexibility and adaptability, as the technologies of studying and researching paper are developing, and with them the volumes of information necessary to store one object are increasing. For a full-fledged analysis, each object undergoes a lot of historical, restoration and natural science research and, so that all information is stored securely and structured, a database is created in SQL using the PostgreSQL database management system. The database contains historical information, photographs and calculations, information about previous restoration, images and spectrum files, quantitative analysis results and many other characteristics. For the comfortable use of all specialists, the database is hosted on the site using the Django framework running in Python.

Keywords: paper, watermark, database, DBMS, PostgreSQL, Django, Admin 4, models, migrations, views, HTML template, Jinja template engine.

References

[1] Bogdanov A.P. Osnovy filigranovedenija. Istorija, teorija, praktika [Fundamentals of watermarks science. History, theory, practice]. Moscow, 1999. 326 s.
[2] Balachenkova A. P., Tsypkin D. O. (2017). Vozmozhnosti tehnologicheskogo analiza istoricheskih bumag v istochnikovedcheskom issledovanii pamjatnikov [Possibilities of technological analysis of historical papers in source-study investigation of monuments]. Vestnik Sankt-Peterburgskogo universiteta. Istorija — [Vestnik of Saint Petersburg University. History]. Vol. 62. Is. 2. 375–399.
[3] Esipova V.A. K voprosu o sozdanii jelektronnyh baz dannyh filigranej [On the creation of electronic databases of filigrees]. Arheograficheskij ezhegodnik — [Archeographic Yearbook]. 2000. 70-93.
[4] Karpova I.P. (2013). Bazy dannyh. Kurs lekcij i materialy dlja prakticheskih zanjatij. Uchebnoe posobie [Databases. A course of lectures and materials for practical classes. Study guide.]. Piter Publ. 240 p.
[5] Sushkov A., Klassen N. (2023). SUBD PostgreSQL: pochemu ejo stoit vybrat' dlja raboty s dannymi i kak ustanovit' [PostgreSQL DBMS: why you should choose it for working with data and how to install it]. Blog Jandeks Praktikuma – [Yandex Practicum Blog]. Available at: https://practicum.yandex.ru/blog/chto-takoe-subd-postgresql/#id1
[6] Django. SkillFactory.Blog (2022). Available at: https://blog.skillfactory.ru/glossary/django/ (Accessed date: 12/10/2022)
[7] Konjuhov V.G. (2017). Baza dannyh. Ponjatie, znachenie i rol' v sovremennom mire [Database. Concept, meaning and role in the modern world]. Sistemnye tehnologii – [System technologies]. No. 24. p. 61-63.

[8] Bernstein. Memory of Paper. Available at: https://www.memoryofpaper.eu/BernsteinPortal/appl_start.disp (accessed date: 24/3/2023).
[9] PostgreSQL: The World's Most Advanced Open Source Relational Database (2022). Available at: https://www.postgresql.org/ (Accessed date: 10/10/2022)
[10] Fors'e Dzh., Bisseks P., Chan U. (2009). Django. Razrabotka veb-prilozhenij na Python. [Django. Development of web applications in Python.]. Translated from English. SPb. Simvol-Pljus Publ. 456 p.
[11] Django ORM Cookbook. DJANGO.Fun (2019). Available at: https://django.fun/en/docs/django-orm-cookbook/2.0/ (Accessed date: 25/10/2022)
[12] PgAdmin 4 Documentation. Release 6.20 (2023). Available at: https://www.pgadmin.org/docs/pgadmin4/6.21/index.html (Accessed date: 03/03/2023)
[13] Novikov B.A., Gorshkova E.A., Grafeeva N.G. (2020). Osnovy tehnologij baz dannyh. Uchebnoe posobie. 2e izd. [Fundamentals of database technologies. Study guide. 2nd edition]. DMK Press Publ. 582 p.
[14] Rusakov M. (2022). Django View – bazovye predstavlenija [Django View – Basic views]. Pomoshhnik Python – [Python Helper]. Available at: https://pythonpip.ru/django/django-view-bazovye-predstavleniya (Accessed date: 15/10/2022)
[15] Get started with Bootstrap. GitHub (2022). Available at: https://github.com/twbs/bootstrap/blob/v5.3.0-alpha1/site/content/docs/5.3/getting-started/introduction.md (Accessed date: 15/12/2022)