Государственное бюджетное общеобразовательное учреждение

города Москвы «Школа № 1474»

**Парный итоговый проект**

**«Сайт общеобразовательных учреждений Москвы»**

Выполнили: Новоселова Дарья, 10 «Н» класс, Шипкова Виктория, 10 «П» класс

Руководитель: Костяев Филипп Александрович, учитель информатики

Москва, 2023 г.

**Содержание**

Оглавление

[Введение 3](#_Toc125576856)

[1. Теория разработки сайтов 5](#_Toc125576857)

[1.1 Способы разработки сайтов 5](#_Toc125576858)

[1.1.1 Разработка с помощью конструктора 5](#_Toc125576859)

[1.1.2 Разработка с помощью CMS 6](#_Toc125576860)

[1.1.3 Разработка с нуля 6](#_Toc125576861)

[1.2 Сравнение фреймворков Django и Flask 7](#_Toc125576862)

[2. Создание сайта 11](#_Toc125576863)

[2.1 Парсинг сайтов 11](#_Toc125576864)

[2.2 Создание сайта 16](#_Toc125576865)

[Заключение 22](#_Toc125576866)

# Введение

**Актуальность:** Получение качественного основного общего образования – одно из условий успешности человека в современном мире. Общеобразовательные учреждения закладывают в учеников базовые знания по различным дисциплинам, являются одним из способов социализации, а также, несомненно, готовят к взрослой жизни и, в частности, помогают определиться с будущей профессией.

В наше время довольно сложно найти свое призвание, ведь существует огромное количество специализаций. Школы Москвы предоставляют возможность ученикам попробовать себя в разных направлениях и выбрать то, которое приглянется больше всего. Для этого во многих школах есть предпрофильные классы, в которых углубленно изучаются какие-либо предметы, а также множество секций, которые помогут реализовать свой творческий потенциал, дополнительно изучить предмет или познакомиться с другими учениками школы. Многие общеобразовательные учреждения также сотрудничают с вузами, что дает некоторые дополнительные возможности для школьников.

Официальные сайты школ содержат одинаковую по структуре информацию, но поиск по ним с целью выбора учебного заведения займет большое количество времени и крайне неудобен. Наш проект поможет структурировать все эти данные по категориям (районы, доступные кружки, предпрофильные классы, сотрудничество с вузами) и отсортировать их по рейтингу. Это упростит школьникам и их родителям выбор учебного заведения.

**Цель работы:** создать сайт с базой данных общеобразовательных учреждений города Москвы

**Задачи:**

* проанализировать способы создания сайтов и подобрать наилучшую технологию;
* изучить необходимые инструменты для создания сайта;
* изучить основы парсинга сайтов для получения существующей информации на сайтах школ;
* изучить доступную на сайтах московских школ информацию и собрать ее в единую базу данных;
* реализовать все этапы создания сайта.

# Теория разработки сайтов

## 1.1 Способы разработки сайтов

Существует множество способов создания сайтов, но все их можно разделить на три категории:

* разработка с помощью конструктора;
* разработка с помощью CMS;
* разработка самостоятельно с нуля (в частности с использованием фреймворков).

Рассмотрим каждый из этих способов.

### 1.1.1 Разработка с помощью конструктора

Конструктор сайта – инструмент, позволяющий создавать сайты без знания языков программирования. Конструктор предлагает построить сайт по модульному принципу из уже готовых частей. Такой способ подойдет для небольших сайтов.

Плюсы:

* Низкая цена. Практически все конструкторы бесплатны;
* Простота использования. Для разработки сайта с помощью конструктора не нужно иметь особых навыков;
* Вся рутина делается конструктором. Различные процессы (размещение на хостиге, подключение модулей) осуществляются через панель управления несколькими нажатиями на кнопки.

Минусы:

* Скрытые затраты. Возможны дополнительные затраты: размещение на хостинге, электронная почта с именем домена и другое;
* Тяжеловесность сайта. Конструктор содержит в себе много программного кода, не относящегося к самому сайту, но необходимого для отображения его внешнего вида. Из-за этого сайт, сделанный на конструкторе будет загружаться значительно дольше сайта, сделанного вручную или с помощью CMS;
* Отсутствие доступа к программному коду. Нет возможности изменить или дополнить сайт какими-то функциями.

### 1.1.2 Разработка с помощью CMS

CMS (Content Managment System) — система создания и управления сайтом. Позволяет управлять его контентом и хранить данные пользователей. Существуют различные плагины и модули, с помощью которых можно расширять возможности сайта.

Плюсы:

* Бесплатный доступ. Почти все CMS изначально бесплатны;
* Удобное управление контентом. При использовании CMS не требуется особых навыков для управления контентом;
* Множество готовых решений.

Минусы:

* Уязвимость сайта, плохо защищены от взлома;
* Требования к знаниям. Могут потребоваться знания по программированию (чаще всего на PHP) и верстке, если нужно расширить какой-то модуль или написать проект с нуля;
* Затраты на дополнительный контент. Конечно, существует множество бесплатных дополнений, но качество таких модулей чаще всего не самое лучшее;
* Большое количество ненужных функций «из коробки» (установленных по умолчанию). CMS изначально содержит множество возможностей, далеко не все из которых понадобятся для конкретного сайта, но будут замедлять его.

### 1.1.3 Разработка с нуля

Самостоятельная разработка осуществляется с помощью различных инструментов и позволяет создать совершенно любой проект с различными функциями.

Плюсы:

* Создание любых функций и дизайна;
* Широкие возможности продвижения. В отличие от CMS и конструкторов, продвинуть в естественном поиске самостоятельно разработанный сайт намного легче.

Минусы:

* Наличие знаний. Разработка с нуля требует наличия большого количества знаний по языкам программирования, построению архитектуры веб-приложений, алгоритмам и структурам данных;
* Временные затраты.

**Вывод:** Разработка с нуля позволяет реализовать любой функционал для сайта, и интересна в учебных целях, поэтому наш сайт будет создан таким способом.

## 1.2 Сравнение фреймворков Django и Flask

Наш сайт будет написан с использованием языка программирования *Python*.

*Python* является одним из наиболее популярных языков программирования для веб-разработки. Это объясняется рядом причин:

* Python – интерпретируемый язык, поэтому он не зависит от платформы. К тому же, программы, написанные на интерпретируемых языках более гибкие, они используют динамическую типизацию (переменная связывается с типом не при объявлении переменной, а в момент присваивания значения; благодаря этому в различных участках программы переменная может принимать значения разных типов) и имеют небольшой размер;
* Python – объектно-ориентированный язык программирования. Это упрощает обновление кода, делает программу читабельнее, позволяет эффективнее работать в команде над одним проектом;
* Простотой синтаксис, большое количество библиотек для разных задач (например, для подключения базы данных или выполнения сложных расчетов);
* Совместимость с различными фреймворками.

Для создания сайтов часто используют фреймворки.

Фреймворк (англ. framework – «каркас») – платформа для упрощения создания и работы над программным проектом. Формирует структуру программной системы. Во фреймворке реализованы классы, определены переменные и константы, созданы готовые решения различных проблем (обработка запросов, работа с базами данных, формами).

Существует множество различных фреймворков на Python, но остановимся на двух наиболее популярных: *Django* и *Flask*.

**Django** - высокоуровневый веб-фреймворк с открытым исходным кодом. Используется популярными веб-приложениями и сайтами, такими как Mozilla, Instagram, Dropbox, Pinterest, Spotify.

**Flask** – микрофреймворк (упрощенный фреймворк) с минималистичными функциями. Следующие сайты компаний написаны на *Flask*: Red Hat, Reddit, Netflix, Airbnb, RackSpace, Lyft.

**Возможности фреймворков**

**Формы**

Форма – раздел HTML-документа, в котором хранится введенная пользователем информация для последующей обработки.

*Django* предлагает инструменты для работы с формами. Поддерживается создание форм, проверка данных и CSRF-токенов (уникальные значения веб-приложений, предотвращающие подделку межсайтовых запросов). Формы отображаются при помощи встроенных шаблонов.

*Flask* не имеет встроенных инструментов для работы с формами. Для использования форм существует библиотека Flask-WTF.

**Обработка http-запросов и маршрутизация**

Http-запрос – сообщение, отправляемое клиентом (пользователем) на сервер, для вызова определенных действий на сайте. Обработка запросов – ключевой момент в любом веб-приложении.

В *Django* обработка запросов происходит отдельно от определения маршрутов (адресов страниц). Когда пользователь запрашивает страницу, *Django* проходит по всем шаблонам URL, сопоставляет их с заданным адресом и вызывает представление страницы, находящееся в другом файле.

Объект request, в котором хранится запрос, подается в *Django* каждый раз в качестве аргумента. В этом случае контекст запроса отделен от самого веб-приложения, но передача параметра каждый раз, когда нужно получить доступ к запросу, может быть неудобной.

Во *Flask* маршруты объявлены с помощью декораторов в одном месте.

Объект request объявлен как локальная переменная потока и доступ к нему можно получить просто импортировав из библиотеки. Значит, передавать запрос не нужно, что значительно облегчает работу.

**Шаблоны**

Шаблон – инструмент, упрощающий работу с разметкой сайта и связывающий ее с данными.

В *Flask* используется шаблонизатор Jinja2. *Django* имеет собственный шаблонизатор Django Template Engine и допускает использование других языков шаблонов.

Оба фреймворка также предоставляют возможность использовать статические файлы в шаблонах.

**ORM**

ORM – инструмент, связывающий базу данных с объектно-ориентированным языком. Т.е. это технология, преобразующая данные в таблице в классы, а классы – в данные.

В *Django* есть ORM, подходящий для большинства случаев, кроме некоторых сложных запросов. В этом случае можно использовать SQLAlchemy. Поддерживается миграция моделей (добавление изменений моделей в базу данных).

Во *Flask* ORM нет, но можно использовать SQLAlchemy. К тому же, *Flask* дает свободу в управлении базами данных, в отличии от *Django*, где многое сделано автоматически.

**Безопасность и производительность**

*Django* поддерживает защиту от XSS, CSRF, SQL-инъекций и кликджекинга.

*Flask* поддерживает защиту от XSS.

*Flask* ввиду своей легкости и малого количества дополнительных функций быстрее *Django*, но на практике эти различия не сильно заметны.

**Вывод:** *Django* имеет больший встроенный функционал, нежели *Flask,* а также больше защищен. Поэтому в нашем проекте будет использован фреймворк Django.

# 2. Создание сайта

## 2.1 Парсинг сайтов

Парсинг сайтов – автоматизированный сбор информации, ее структурирование и обработка.

В нашем проекте парсинг будет использован для сбора информации с портала открытых данных Москвы (<https://data.mos.ru>), чтобы получить список всех школ Москвы, и с сайтов школ, чтобы получить уже конкретные данные по каждой из школ. Парсинг осуществим также на *Python*.

Первым делом спарсим начальную информацию о школах (файл *db/parser\_from\_opendata.py*). Для этого с помощью библиотеки *request* программа отправляет get-запрос на сайт и получает список школ (см. Рисунок1). Далее сохраняем их в удобном формате в базу данных с помощью библиотеки *SQLite3* (см. Рисунок 2).

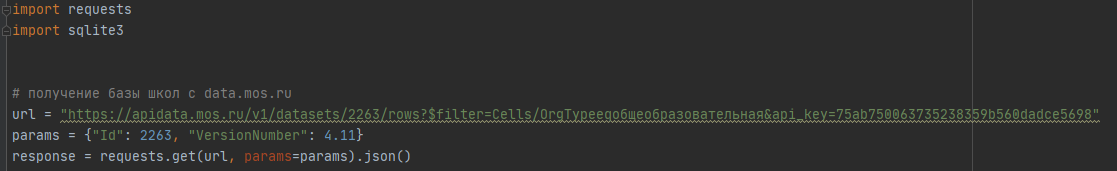


Рисунок 1 - сбор информации с data.mos.ru



Рисунок 2 - обработка данных

В итоге мы получили и сохранили в таблицу schools (Рисунок 3) базы данных полные (full\_name) и краткие (short\_name) названия школ, их сайты (web\_site), образовательные программы (educational\_services), организационно-правовые нормы (legal\_organization), ФИО руководителей (chief\_name), номера телефонов (public\_phone), электронные почты (email) и адреса (institutions\_addresses). Так как каждая школа имеет несколько адресов, для каждого из которых есть своя информация, то их выносим в отдельную таблицу addresses (Рисунок 4), состояющую из района (disctrict), округа (adm\_area), адреса (address), номера телефона (public\_phone) и доступности для инвалидов (available\_k, available\_o, available\_z, available\_s). Структура получившейся базы данных:

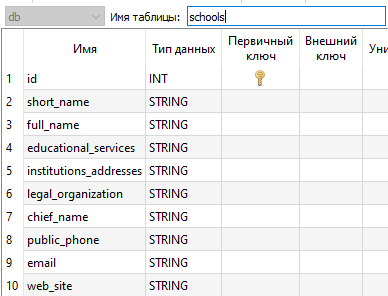


Рисунок 3 - таблица schools

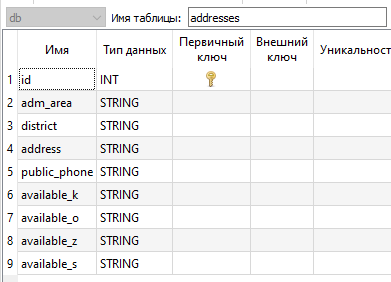


Рисунок 4 - таблица addresses

Следующим шагом соберем данные с сайта каждой школы (файл *db/parser\_from\_sites.py*). Добавим таблицы predprof, sections и universities для предпрофильных классов, секций и университетов соответственно, а также колонки predprof\_id, sections\_id и universities\_id в таблицу schools.

В первую очередь нужно получить информацию обо всех существующих предпрофильных классах в Москве. Воспользуемся библиотекой *Selenium[[1]](#footnote-1)*, чтобы получить доступ к отдельным элементам страницы <https://profil.mos.ru> и записать таблицу predprof базы данных названия и описания предпрофильных классов (Рисунок 5).

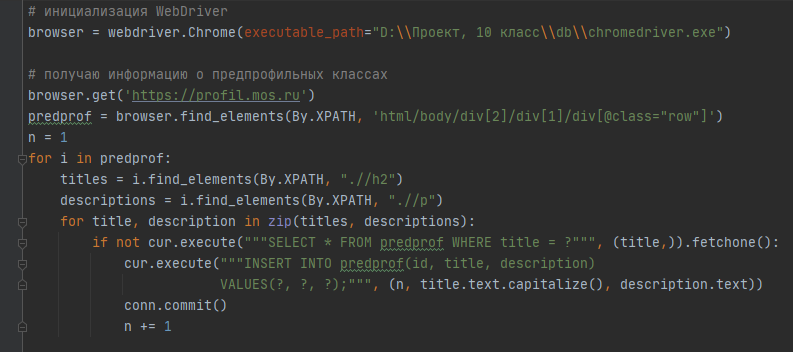


Рисунок 5 – получение общей информации о предпрофильных классах

Далее проходимся по сайтам школ. Сайты школ созданы по одному шаблону, поэтому многие элементы на страницах названы одинаково, что и дает нам возможность автоматизировать процесс сбора данных.

С главной страницы можем получить названия предпрофильных классов конкретной школы. В столбец predprof\_id таблицы schools запишем id предпрофильных классов, которые находятся на главной странице школы, из таблицы predprof (Рисунок 6).

Вузы-партнеры школы получим пройдясь по страницам всех предпрофильных классов (Рисунок 7). Запишем информацию о них (название и количество поступивших выпускников школы в этот вуз) в таблицу universities.

Информация о секциях находится на странице /dop-obr/poisk-kruzhkov-i-sekcij. В таблицу sections поместим название кружка, его категорию (категории секций у разных школ также одинаковые), возрастную категорию, форму обучения (групповая или индивидуальная) и бесплатно ли обучение (Рисунок 8).



Рисунок 6 - предпрофильные классы отдельной школы

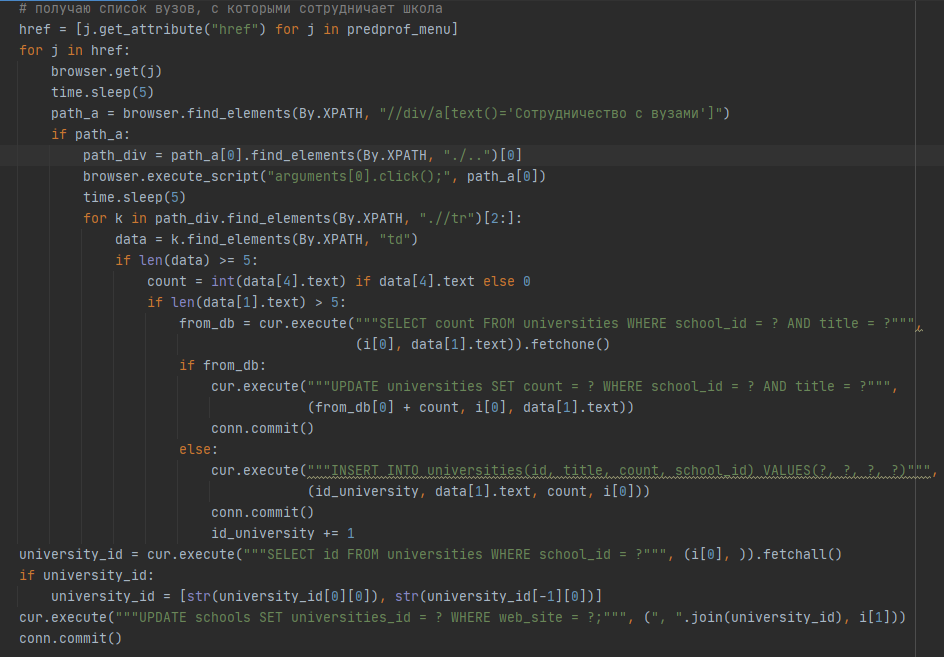


Рисунок 7 - вузы-партнеры отдельной школы

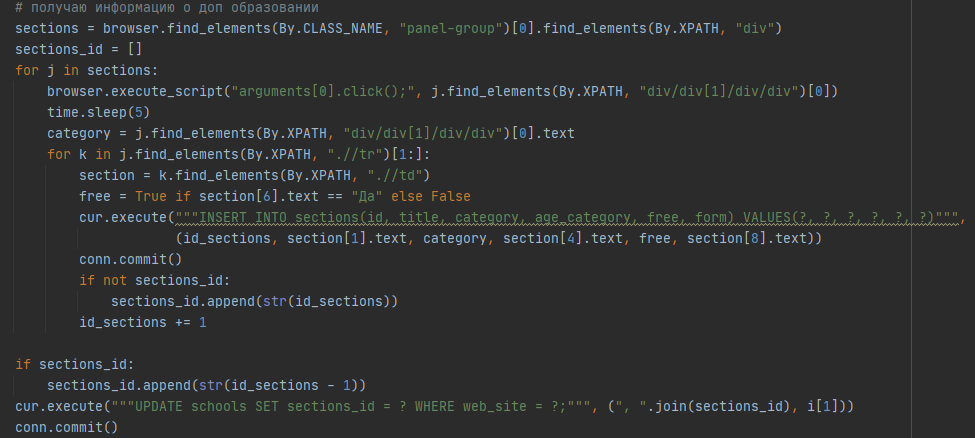


Рисунок 8 - дополнительное образование

На сайтах школ по-разному указаны названия вузов, поэтому создадим еще одну небольшую программу, которая заменит вариации одного названия самым коротким вариантом (Рисунок 9):

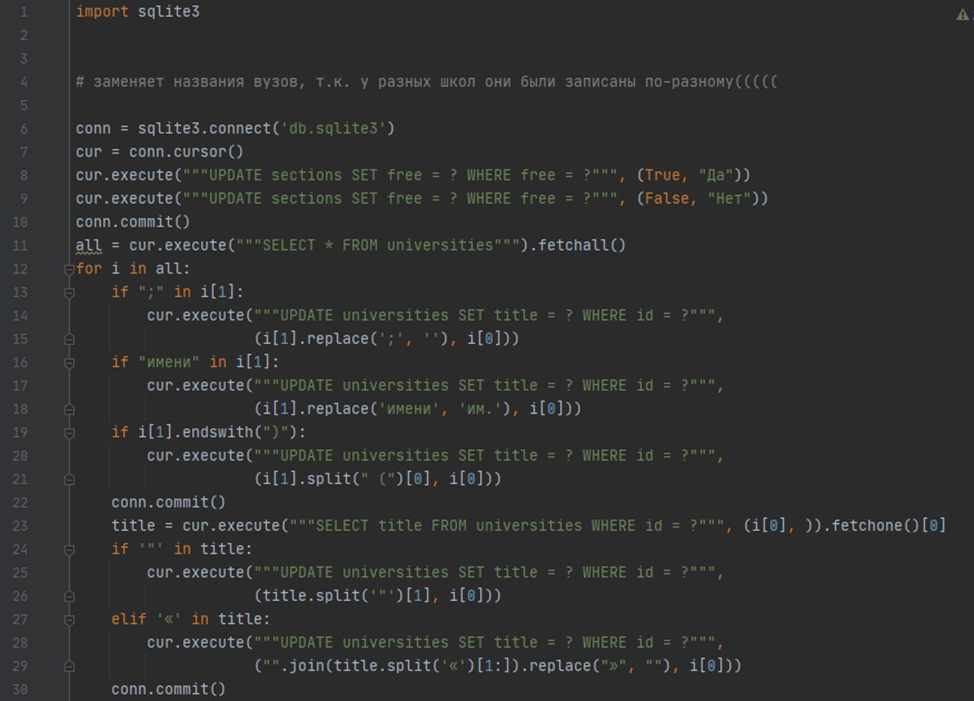


Рисунок 9 - изменение названий вузов

## 2.2 Создание сайта

После заполнения базы данных можно приступать к созданию сайта. Для этого с помощью библиотеки *Django* создаем новое приложение schools\_site в проекте project. В файл project/project/settings.py в переменную INSTALLED\_APPS дописываем наше созданное приложение (Рисунок 10). В том же файле записываем путь к папке со статическими файлами (/static/) в переменную STATIC\_URL. Созданная ранее база данных хранится в папке project. Шаблоны будут хранится в папке project/schools\_site/templates, статические файлы в project/schools\_site/static.

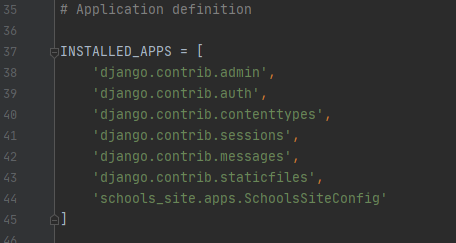


Рисунок 10 - добавление приложения в файл настроек проекта

Так как у нас уже есть база данных, модели (определенные классы, используемые Django, определяющие поля таблиц) для нашего сайта создадим командой **python manage.py inspectdb > models.py**.

В файле project/schools\_site/models.py теперь находится 5 моделей наших таблиц. Напишем также для каждой из них класс <название таблицы>Manager, унаследованный от класса Manager, с методом get\_queryset, возвращающим QuerySet из объектов той или иной модели. Для модели Schools (Рисунок 11) также пропишем методы get\_absolute\_url (возвращает адрес страницы отдельной школы), get\_predprof (возвращает QuerySet предпрофильных классов школы), get\_universities (возвращает QuerySet вузов-партнеров школы), get\_forms (возвращает формы обучения школы), get\_addresses (возвращает QuerySet адресов школы)[[2]](#footnote-2).

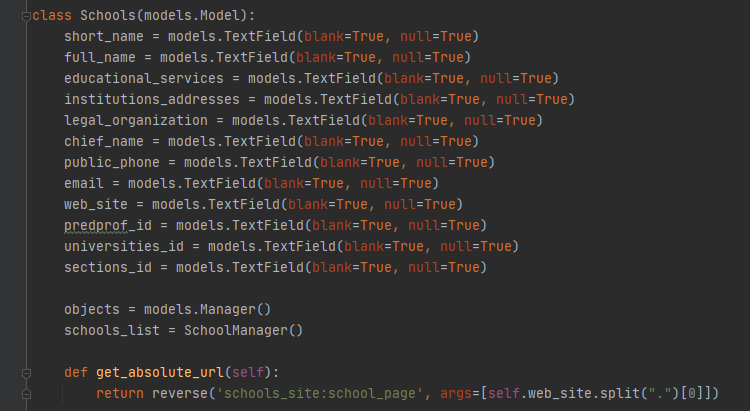


Рисунок 11 - модель Schools

Начальная настройка приложения закончена, после нее приступаем к написанию обработчиков страниц (Рисунок 12). В файле project/schools\_site/views.py определяем функции home(request) и school\_page(request, url). Первый обработчик принимает запрос и возвращает главную страницу, второй принимает запрос и аргумент url, который является обрезанным адресом школы (отброшена часть .mskobr.ru) и возвращает страницу отдельной школы. В шаблон главной страницы из обработчика отправляется QuerySet школ, уникальные значения районов, округов, категорий секций, предпрофессиональных классов, вузов-партнеров и форм обучения, а также номер страницы, на которой находится пользователь (на одной странице отображается 10 школ). В шаблон страницы школы из обработчика отправляется объекты школы, адреса, предпрофессиональных классов, университетов, форм обучения и категорий секций.

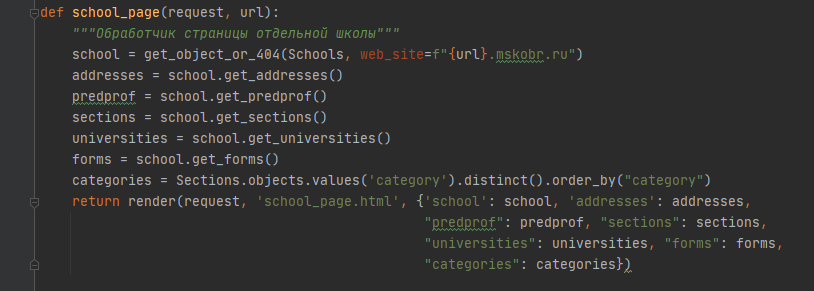


Рисунок 12 - обработчик страницы отдельной школы

В файл project/schools\_site/urls.py добавляем адреса созданных страниц.

Далее создадим разметки страниц сайта. Для удобства сначала напишем базовый шаблон (templates/base.html), от которого будут наследоваться все остальные шаблоны. В нем подключим файл со стилями static/css/style.css и другие необходимые стили. Также пропишем кнопку, ведущую на главную страницу (Рисунок 13).

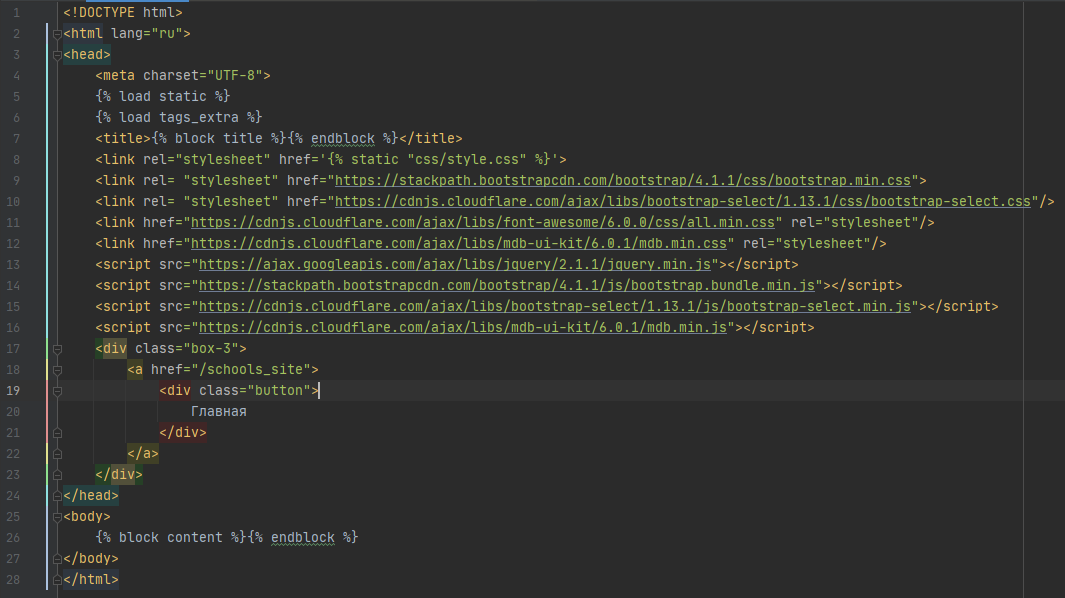


Рисунок 13 - базовый шаблон

Шаблон главной страницы (templates/home.html) будет состоять из фильтров, строки поиска и списка школ. Для отправки изменений в фильтрации мы не будем использовать формы. Вместо этого напишем функцию с помощью библиотеки JQuery на JavaScript, которая отслеживает изменения в фильтрах и переход на следующую страницу главной страницы и отправляет post-запрос, содержащий состояния всех фильтров и строки поиска (Рисунок 14).

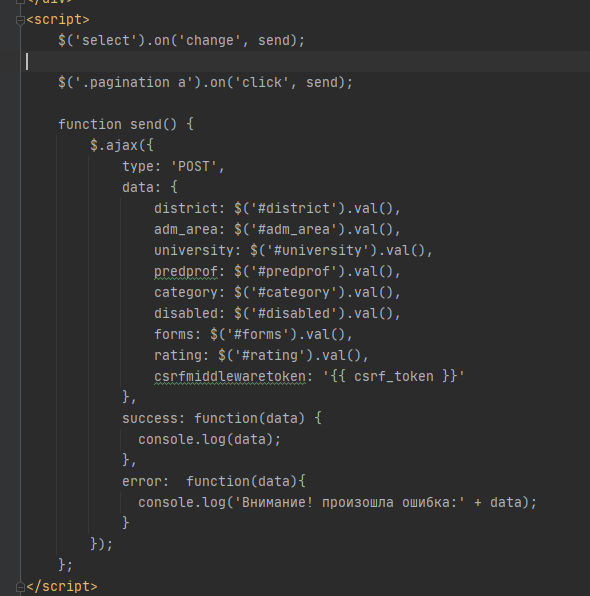


Рисунок 14 - функция для отправки post-запроса в шаблоне главной страницы

Страница школы будет содержать всю информацию, которую нам удалось получить ранее. Схематичное расположение элементов представлено на (Рисунок 17).

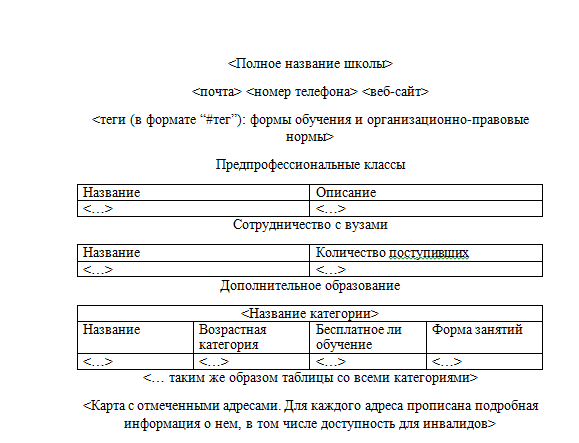


Рисунок 15 - схема страницы школы

В этом шаблоне нам также нужно будет использовать JQuery, чтобы таблички могли сворачиваться и разворачиваться при нажатии на заголовок, так как информации очень много (Рисунок 16). К тому же по мере разработки появилась необходимость в создании дополнительного фильтра in\_category для этого шаблона, чтобы разделить все секции по категориям (Рисунок 17). Этот фильтр прописан в файле project/schools\_site/templatetags/tags\_extra.py. Также его нужно добавить в файл с настройками в переменную TEMPLATES.

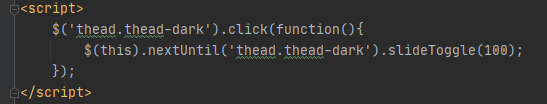


Рисунок 16 - функция для сворачивания таблиц в шаблоне страницы школы



Рисунок 17 - использование пользовательского фильтра в шаблоне страницы школы

# Заключение

Таким образом, в данном проекте мы рассмотрели способы создания сайтов и сравнили два наиболее популярных фреймворка для их создания. Получили опыт написания парсеров и работы с базами данных.

Также мы достигли главной цели проекта – создания сайта. Наш сайт содержит структурированную информацию о школах Москвы и позволяет фильтровать ее по различным критериям.

Наш проект может помочь быстрее найти необходимую информацию о школе и на ее основе сделать выбор учебного заведения.

1. На сайтах могут присутствовать элементы, появляющиеся после совершения определенных действий. Например, информацию в свернутых таблицах можно получить только после нажатия на заголовок таблицы. *Selenium* предлагает удобный функционал для совершения подобных манипуляций, поэтому мы используем именно эту библиотеку. [↑](#footnote-ref-1)
2. Создание этих методов обусловлено тем, что поля таблицы schools не привязаны к полям других таблиц, а только хранят строки с id соответствующих объектов. [↑](#footnote-ref-2)