Государственное бюджетное общеобразовательное учреждение

города Москвы «Школа № 1474»

**ИТОГОВЫЙ ПРОЕКТ**

**«САЙТ-АГРЕГАТОР ОБЩЕОБРАЗОВАТЕЛЬНЫХ УЧРЕЖДЕНИЙ МОСКВЫ»**

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

Руководитель: Костяев Филипп Александрович, учитель информатики

Москва, 2023 г.

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:** 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-агрегатор с базой данных Государственных Бюджетных общеобразовательных учреждений города Москвы

**Задачи:**

* собрать список сайтов всех ГБОУ Москвы;
* собрать в единую базу данных доступную на сайтах московских школ информацию;
* проанализировать способы создания сайтов и подобрать наилучшую технологию;
* реализовать все этапы создания сайта.

# 1 Парсинг сайтов

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

В проекте парсинг будет использован для сбора информации с портала открытых данных Москвы [1], чтобы получить список всех школ Москвы, и с сайтов школ, чтобы получить уже конкретные данные по каждой из школ. Парсинг осуществим на языке программирования Python.

Начальная информация о школах (файл db/parser\_from\_opendata.py). с помощью библиотеки request*.*  Программа отправляет GET-запрос на сайт и получает список школ (рисунок1). Сохраним их в удобном формате в базу данных с помощью библиотеки SQLite3(рисунок 2).

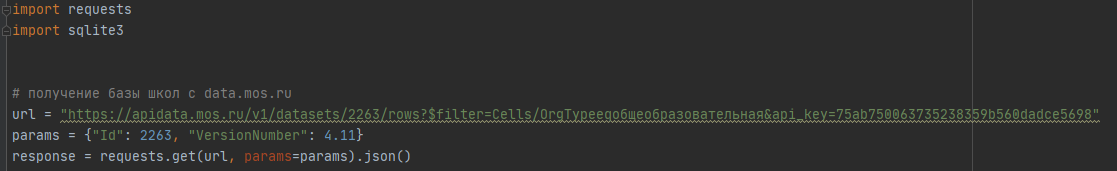


Рисунок 1 - Сбор информации с data.mos.ru



Рисунок 2 - Обработка данных

Структура таблицы schools базы данных (рисунок 3):

* полное (full\_name) и краткое (short\_name) название школы;
* сайт школы (web\_site);
* образовательные программы (educational\_services);
* организационно-правовые нормы (legal\_organization);
* ФИО руководителя (chief\_name);
* номер телефона (public\_phone);
* электронная почта (email);
* адреса (institutions\_addresses).

Так как каждая школа имеет несколько адресов, для каждого из которых существует 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 представлена на рисунке 4.

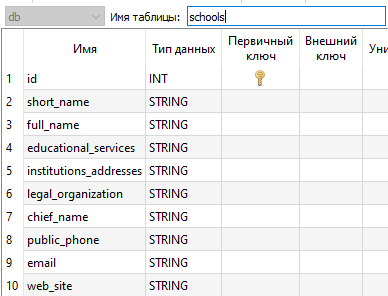


Рисунок 3 - Таблица schools

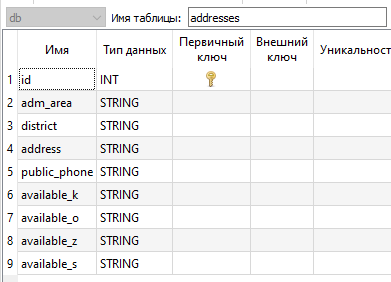


Рисунок 4 - Таблица addresses

Данные, полученные с сайтов каждой школы, находятся в файле 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.

Для получения информации обо всех существующих предпрофильных классах в Москве воспользуемся библиотекой Selenium[[1]](#footnote-1). Необходимую информацию получим с сайта [2] обращением к различным элементам разметки (рисунок 5).

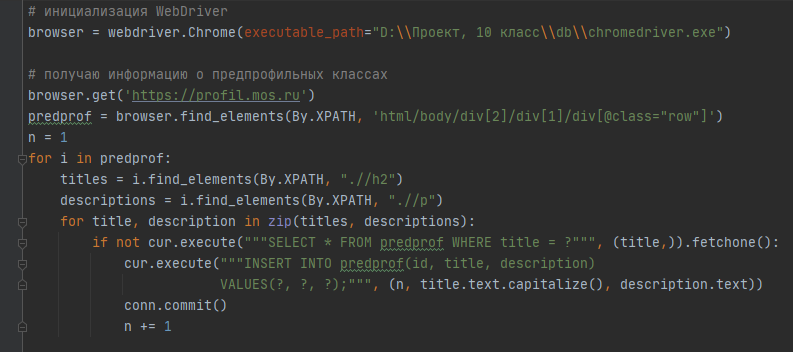


Рисунок 5 – Получение общей информации о предпрофильных классах

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 указаны в разделе каждого предпрофильного класса. На рисунке 7 представлен фрагмент программы, обращающийся к страницам предпрофильных классов и сохраняющий полученные данные (название и количество поступивших в этот вуз выпускников) 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 одинаковые), возрастную категорию, форму обучения (групповая или индивидуальная) и стоимость обучения (рисунок 8). [3]



Рисунок 6 – Получение предпрофильных классов школ

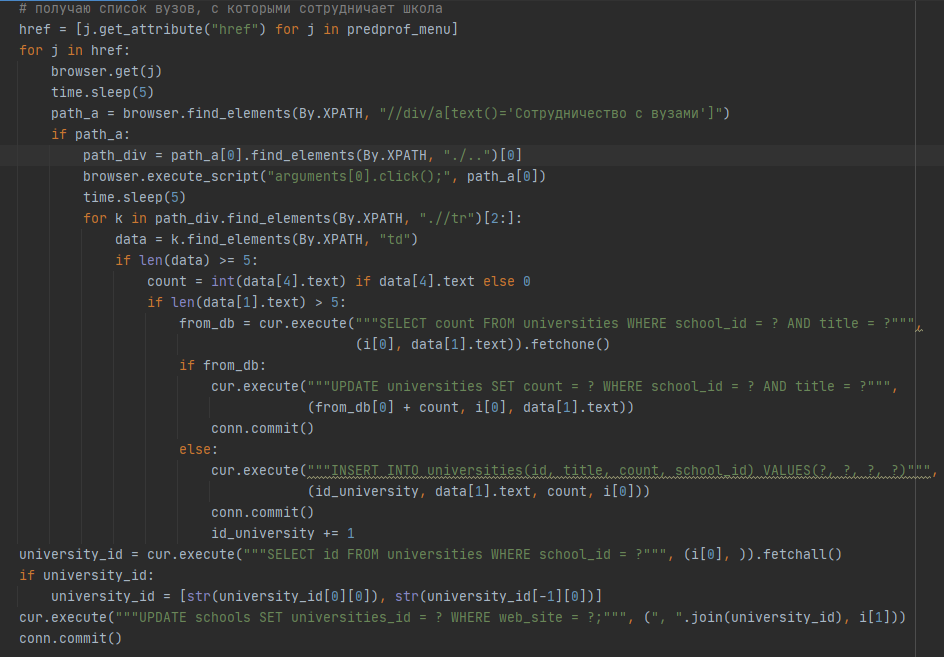


Рисунок 7 – Получение вузов-партнеров школ

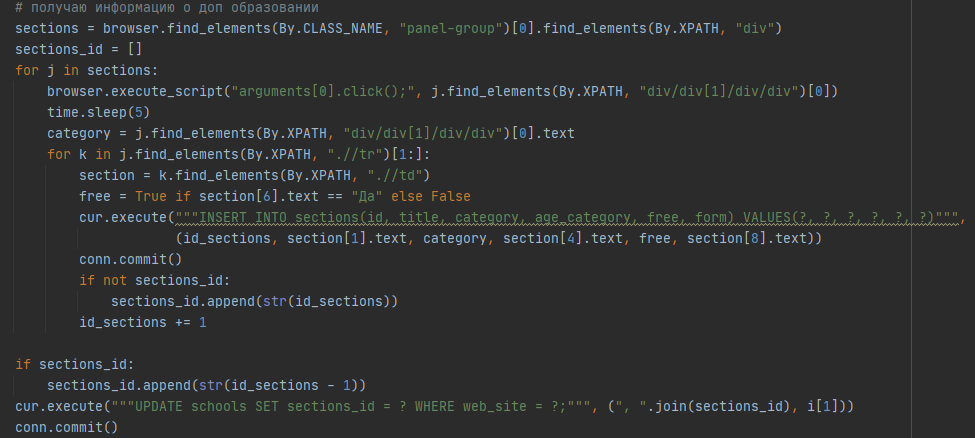


Рисунок 8 – Получение информации о дополнительном образовании

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

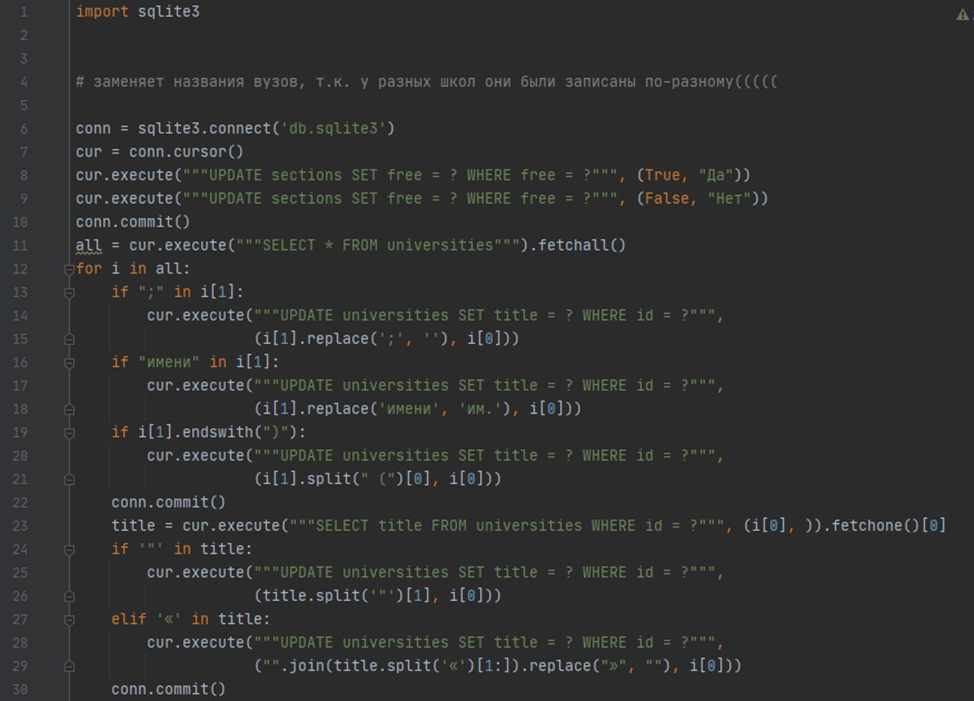


Рисунок 9 - Изменение названий вузов

# Теория разработки сайтов

## 2.1 Способы разработки сайтов

### 2.1.1 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:

* 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:

* 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 гибкости в разработке.

### 2.1.2 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:

* Удобное управление контентом. При использовании CMS не требуется особых навыков для управления контентом;
* Множество готовых решений.

Минусы:

* 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;
* 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.

### 2.1.3 Нативная разработка

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

Плюсы:

* Создание любых функций и дизайна.

Минусы:

* Наличие знаний. Разработка с нуля требует наличия большого количества знаний по языкам программирования, построению архитектуры веб-приложений, алгоритмам и структурам данных;
* Временные затраты. Нативная разработка требует достаточно большого количества времени для реализации продукта.

Обобщая вышесказанное, получим сравнительную характеристику способов разработки сайтов (таблица 1).

Таблица 1 - Сравнение способов разработки сайтов

|  |  |  |
| --- | --- | --- |
|  | Преимущества | Недостатки |
| Конструктор | Простота использования;  Автоматизация рутинных действий | Тяжеловесность сайта;  Отсутствие гибкости в разработке |
| CMS | Удобное управление контентом;  Множество готовых решений | Требования к знаниям;  Ненужные функции, установленные по умолчанию |
| Нативная разработка | Создание любых функций и дизайна | Требования к знаниям;  Временные затраты |

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

## 2.2 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.

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.

Подводя итоги получим таблицу 2.

Таблица 2 - Сравнение фреймворков Django и Flask

|  |  |  |
| --- | --- | --- |
|  | Django | Flask |
| Формы | Удобный функционал для создания и обработки форм | Встроенных инструментов нет |
| Обработка http-запросов | Отдельно от маршрутов | Маршруты объявлены с помощью декораторов в одном месте |
| Шаблоны | Собственный шаблонизатор; возможность использовать статические файлы в шаблонах | Шаблонизатор Jinja2; возможность использовать статические файлы в шаблонах |
| ORM | Встроенный функционал | Встроенного функционала нет; можно использовать SQLAlchemy |
| Безопасность | Защита от XSS, CSRF, SQL-инъекций кликджекинга | Защита от XSS |

**Вывод:** Django имеет больший встроенный функционал, нежели Flask, что значительно упрощает процесс разработки. В проекте будет использован фреймворк Django.

# Создание сайта

## 3.1 Начальная настройка приложений

С помощью библиотеки Django создадим новые приложения schools\_site и users в проекте project. [4]

Приложение schools\_site отвечает за обработку запросов на главную страницу и страницы школ. Содержит модели школ, адресов, секций, вузов и предпрофильных классов.

Приложение users содержит модели пользователей, отзывов и профилей, а также необходимые формы авторизации, регистрации, добавления отзывов и обновления профиля.

В файл project/project/settings.py в переменную INSTALLED\_APPS дописываем созданные приложения и приложение crispy\_forms для отображения форм (рисунок 10). В том же файле записываем путь к папке со статическими файлами (/static/) в переменную STATIC\_URL. В переменных LOGIN\_REDIRECT\_URL, LOGOUT\_REDIRECT\_URL, LOGIN\_URL хранятся пути для переадресации после авторизации, выхода и путь авторизации соответственно.

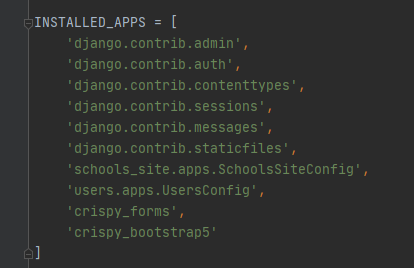


Рисунок 10 - Добавление приложений в файл настроек проекта

В файле project/urls.py (рисунок 11) объявляем пути к регистрации, авторизации, выходу из аккаунта, профилю, добавлению отзывов и страницам, обрабатывающимся приложением schools\_site.

В файле schools\_site/urls.py (рисунок 12) содержатся пути к главной странице сайта и страницам отдельных школ.

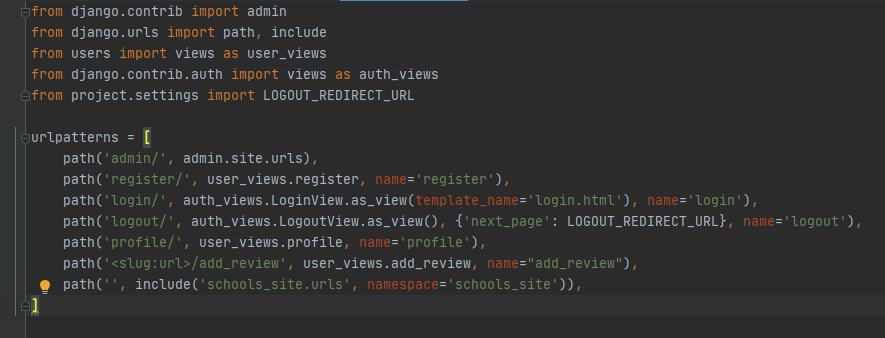


Рисунок 11 - Маршруты проекта

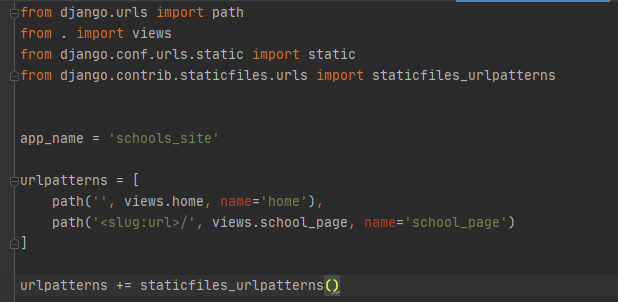


Рисунок 12 - Маршруты приложения schools\_site

## Создание моделей и форм

База данных, созданная в главе 1 Парсинг сайтов, хранится в файле db.sqlite3. Модели (классы, определяющие поля таблиц базы данных) для приложения schools\_site можно создать из существующей базы данных автоматически, выполнив в командной строке команду:

python manage.py inspectdb > models.py

В файле project/schools\_site/models.py теперь находится 5 моделей наших таблиц. Напишем 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 13) 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\_sections (возвращает QuerySet дополнительных кружков);
* get\_addresses (возвращает QuerySet адресов школы)[[2]](#footnote-2).

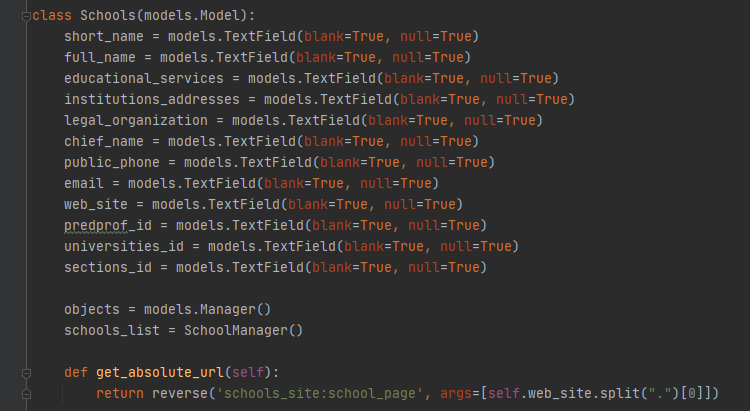


Рисунок 13 – Часть модели Schools

Для приложения users создадим модели Reviews и Profile (классы, унаследованные от django.db.models.Model) для отзывов школ и профилей пользователей соответственно. Модель пользователей User уже создана в Django, она будет использована для формы регистрации и как внешний ключ других моделей.

Модель Reviews включает в себя поля:

* user\_id – id пользователя, который оставил отзыв. Внешний ключ – поле id в модели User;
* school\_id – id школы, для которой был оставлен отзыв. Внешний ключ – поле id в модели Schools приложения schools\_site;
* text – текстовое поле для текста отзыва;
* rating – оценка, целое число.

Модель Profile состоит из:

* user\_id – id пользователя, который оставил отзыв. Внешний ключ – поле id в модели User;
* school\_id – id школы, для которой был оставлен отзыв. Внешний ключ – поле id в модели Schools приложения schools\_site;
* role – роль пользователя (преподаватель, родитель, учащийся)

На рисунке 14 представлена реализация моделей Reviews и Profile.

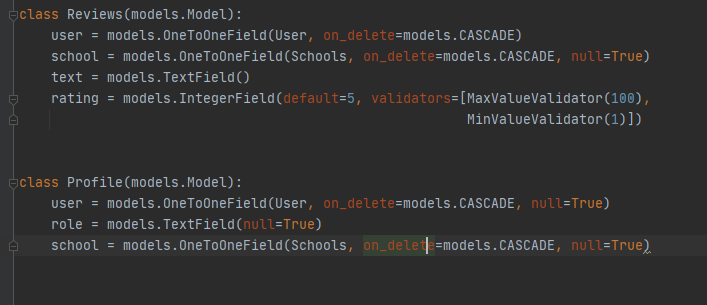


Рисунок 14 - Модели Reviews и Profile

Помимо моделей в приложении users необходимо создать формы, которые впоследствии будут использованы для различных взаимодействий пользователя с сайтом.

Форма UserRegisterForm унаследована от django.contrib.auth.forms.UserCreationForm и служит формой для регистрации пользователя. Форма содержит поля username, email, password1, password2 для имени, почты, пароля и подтверждения пароля соответственно. После сохранения формы ее данные отправляются в модель User.

Форма ReviewForm унаследована от django.forms.ModelForm и служит формой для добавления отзыва школе. Содержит поля text и rating для текста отзыва и оценки соответственно. После сохранения формы данные отправляются в модель Reviews

Форма ProfileForm также унаследована от django.forms.ModelForm и служит для изменения профиля пользователя. Содержит поля role и school\_id для роли пользователя и id выбранной им школы (где он учится, работает или в которой учится его ребенок). Сохраненные данные формы отправляются в модель Profile.

## Создание обработчиков http-запросов

### 3.3.1 Обработчики приложения schools\_site

В приложении schools\_site будут обрабатываться запросы главной страницы (функция home(request)) и страниц школ (функция school\_page(request, url)). Обработчики запросов находятся в файле schools\_site/views.py.

Главная страница содержит фильтры, строку поиска, кнопки авторизации и регистрации (если пользователь авторизирован – кнопки профиля и выхода из аккаунта), кнопки перехода между страницами выданных результатов.

Обработчик главной страницы может получить POST и GET запросы. POST-запрос может быть получен при применении фильтров или переходе на следующую страницу. Изначально для осуществления перехода между страницами был создан объект Paginator из django.core.paginator, но по ходу работы он был заменен на изменение страниц вручную. Это было сделано из-за того, что Paginator отправляет GET-запрос при переходе между страницами, что затрудняет сохранение фильтрации поиска.

Для обработки GET-запроса происходит поиск уникальных значений районов, округов, предпрофильных классов, категорий секций, вузов-партнеров и форм обучения для отображения их в фильтрах. В качестве школ передаются все школы (фильтрация, поиск и переход по страницам обрабатываются при POST-запросе). Результат возвращается с помощью функции render (из модуля django.shortcuts, возвращает шаблон страницы). В шаблон передаются все данные для фильтров, список школ, номера текущей, предыдущей и следующей страниц. На рисунке 15 представлен поиск уникальных значений для фильтров.

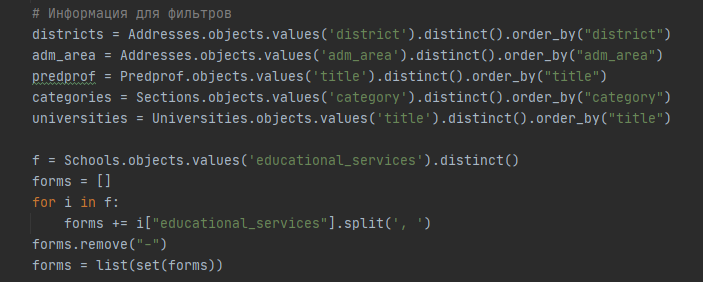


Рисунок 15 - Поиск уникальных значений для фильтрации

Для обработки POST-запроса происходят те же действия, что и при обработке GET-запроса и некоторые дополнительные манипуляции. Из переданного обработчику аргумента request получаются данные POST-запроса. Фильтры в таком случае хранятся как список строк, строка поиска и номер страницы (если был совершен переход) – как строки. Если был совершен POST-запрос из всех школ остаются только соответствующие фильтрам и поисковому запросу (поиск осуществляется по полному и краткому именам школ) школы. Обработчик в таком случае вернет не привычную страницу с помощью render, а ответ в виде JsonResponse. Это необходимо, потому что POST-запрос отправляется не средствами django, а через jQuery.ajax() (в этом случае render не срабатывает, поэтому возвращать его бесполезно). Подробнее об отправке POST-запроса написано в главе 3.4.2 Шаблон главной страницы home.html. На рисунке 16 представлен фрагмент кода с отправкой JsonResponse.

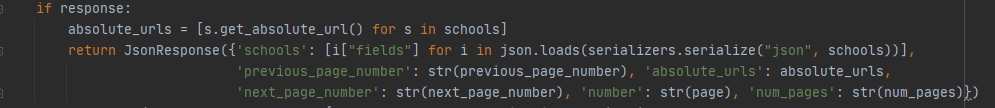


Рисунок 16 - Отправка ответа при POST-запросе на главную страницу

В обработчик страниц отдельных школ POST-запросы не отправляются. Этот обработчик принимает условный url-адрес школы, который создан из адреса сайта школы путем отбрасывания части «.mosobr.ru». По нему можно найти объект школы. Обработчик передает в шаблон объект школы, ее адреса, предпрофильные классы, секции, вузы-партнеры, отзывы и средний рейтинг (вычисляется с точностью до двух знаков после запятой). Реализация обработчика представлена на рисунок 17.



Рисунок 17 - Обработчик страницы отдельной школы

### 3.3.2 Обработчики приложения users

В приложении users реализованы обработчики страниц регистрации (register(request)), добавления отзыва (add\_review(request, url)) и профиля (profile(request)). Обработчики находятся в файле users/view.py.

Обработчик register(request) при POST-запросе создается форма UserRegisterForm со значениями полей, полученных из запроса. При корректно введенных данных создается новый пользователь и происходит перенаправление на главную страницу. При GET-запросе создается пустая форма и передается в шаблон register.html.

Обработчик add\_review(request, url) вторым аргументом получает url-адрес школы (обрезанный адрес настоящего сайта школы), по которому происходит ее поиск. При POST-запросе корректно введенные данные сохраняются в форму и создается соответствующий объект модели Review. Также в модель Review сохраняются поля user\_id и school\_id и происходит перенаправление обратно на страницу школы. При GET-запросе в шаблон add\_review.html передается пустая форма.

Обработчик profile(request) при POST-запросе также сохраняет в форму введенные данные. Если профиль пользователя еще не был создан, он создается. Происходит перенаправление на главную страницу. При GET-запросе создается не пустая форма, как ранее, а заполненная информацией из полей профиля текущего пользователя. В шаблон profile.html передается роль пользователя, id выбранной им школы и список всех школ.

## 3.4 Создание шаблонов

### 3.4.1 Базовый шаблон base.html

Базовый шаблон, а также шаблоны главной страницы и страницы школы хранятся в папке schools\_site/templates, шаблоны страниц авторизации, регистрации, профиля и добавления отзыва – в папке users/templates, стили и изображения – в папке schools\_site/static, пользовательские фильтры – в папке schools\_site/templatetags.

Для удобства сначала напишем базовый шаблон, от которого будут наследоваться все остальные шаблоны. В нем подключим файл со стилями css/style.css и другие необходимые стили. Также пропишем кнопки, ведущие на главную страницу, страницы регистрации и авторизации. Если пользователь авторизирован, вместо кнопок авторизации и регистрации создаются кнопки профиля и выхода из аккаунта. Фрагмент базового шаблона представлен на рисунке 18.



Рисунок 18 - Кнопки на базовом шаблоне

### 3.4.2 Шаблон главной страницы home.html

Шаблон главной страницы (templates/home.html) будет состоять из фильтров, строки поиска и списка школ. Следует заметить важную особенность этого шаблона: в нем будет два блока со школами. Один будет отображаться по умолчанию, другой будет скрыт до отправки POST-запроса.

Для отправки изменений в фильтрации формы использоваться не будут. Вместо этого напишем функции send\_form (рисунок 19) и send\_paginator с помощью библиотеки JQuery на JavaScript [5], которые отслеживают изменения в фильтрах и переход на следующую страницу главной страницы соответственно и отправляют POST-запрос, содержащий состояния всех фильтров и строки поиска. При успешном выполнении запроса вызывается функция load\_html (рисунок 20), которая скрывает блок школ по умолчанию (в него школы больше не будут загружаться) и показывает второй блок. В него добавляются соответствующие строки html-кода.

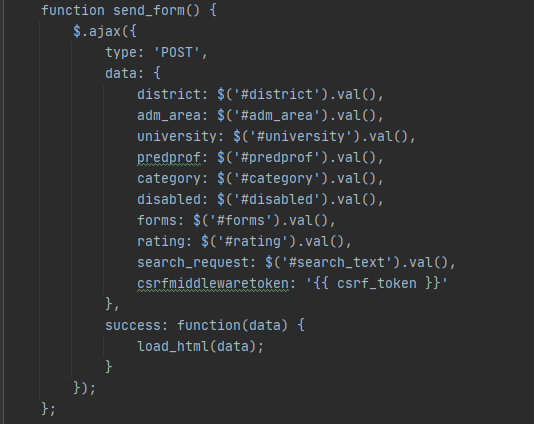


Рисунок 19 - Фунция send\_form в шаблоне home.html



Рисунок 20 - Функция load\_html в шаблоне home.html

На рисунке 21 представлено отображение главной страницы.

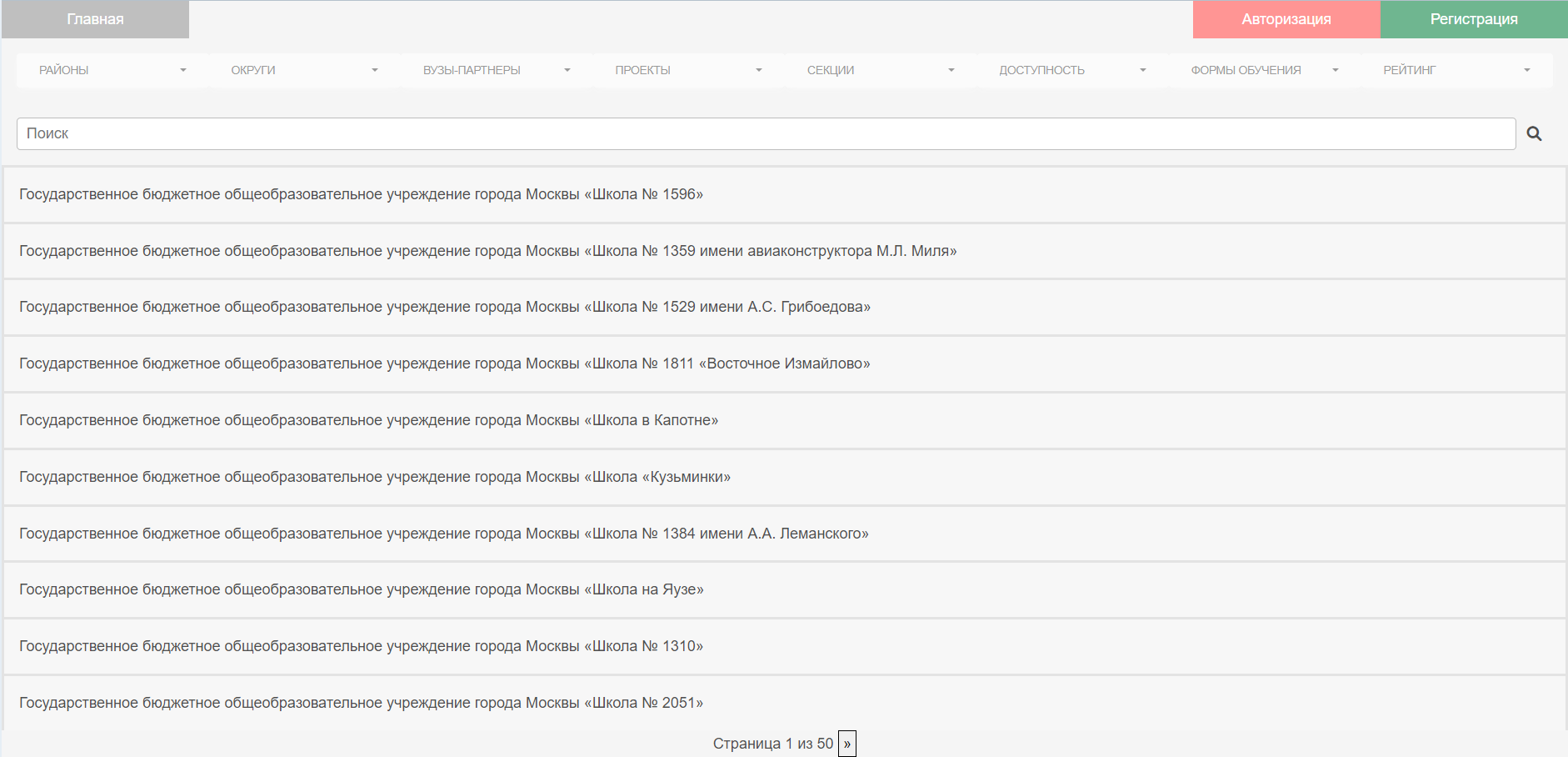


Рисунок 21 - Главная страница

### 3.4.3 Шаблон страницы школы

Страница школы будет содержать всю информацию, которую нам удалось получить ранее. Схематичное расположение элементов представлено на (рисунок 22). Кнопка добавления отзывов доступна, если пользователь авторизирован и не оставлял отзывов к этой школе.

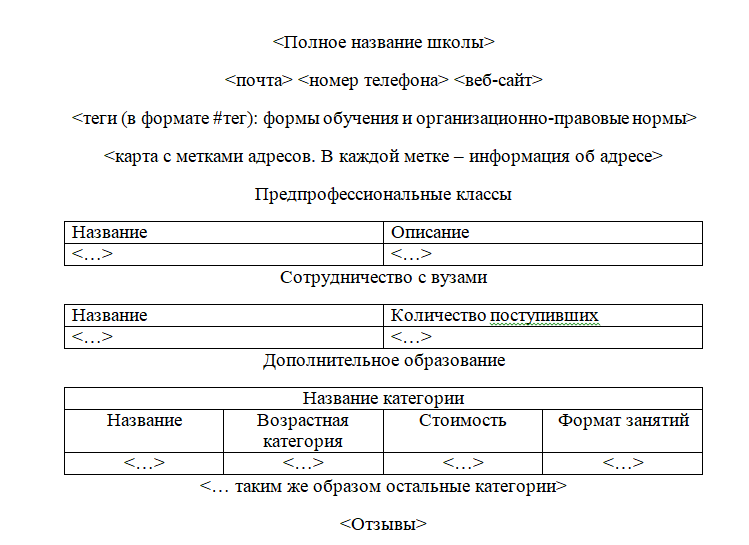


Рисунок 22 - Схема страницы школы

При загрузке страницы в скрипте jQuery сохраняются переменные, содержащие различную информацию об адресах школы: адрес, район, округ, номер телефона и доступность для инвалидов. Они сохраняются отдельно, потому что в скрипте jQuery нет доступа к переменным шаблона.

С помощью этих переменных осуществляется построение карты. Карта строится через JavaScript API Яндекс.Карт. На рисунке 23 представлена часть реализации отображения карты, а на рисунке 24 представлено сохранение информации об адресах в переменные.

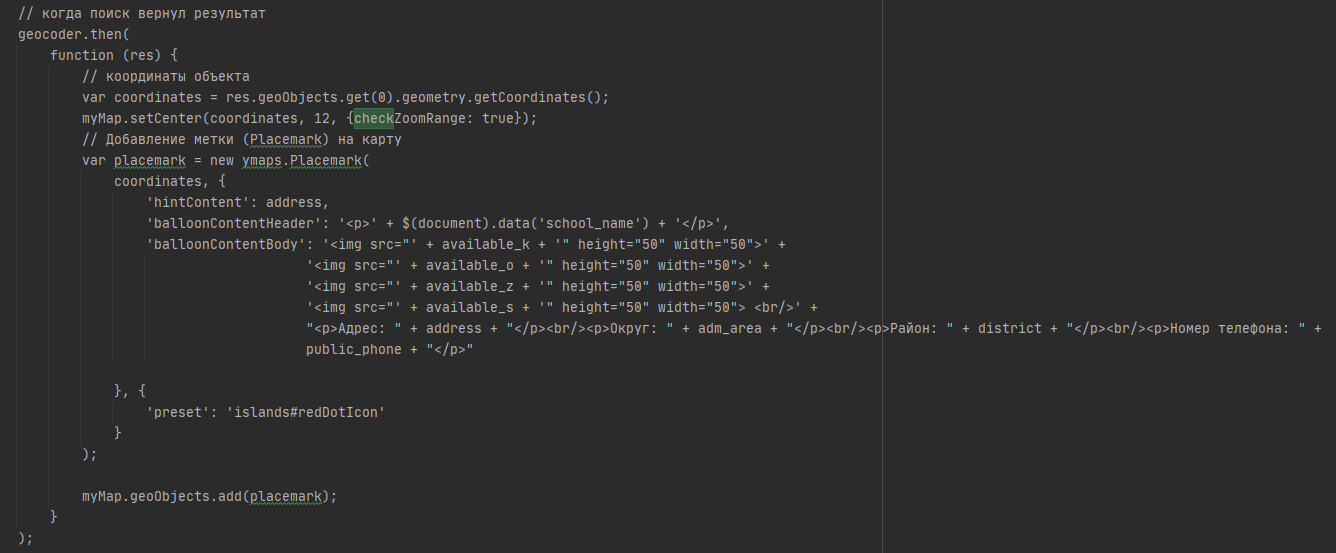
****

Рисунок 23 - Получение объекта geoObject для добавления метки на карте

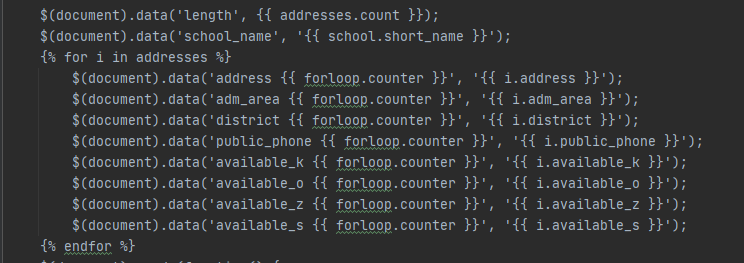
****

Рисунок 24 - Создание переменных с информацией об адресах школы

В этом шаблоне также были использованы пользовательские фильтры. Например, фильтр check\_unique:school для проверки, оставлял ли пользователь отзыв к этой школе (рисунок 25).

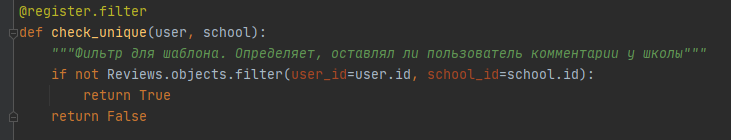


Рисунок 25 - Пользовательский фильтр check\_unique

На рисунке 26 представлено отображение страницы школы. На рисунке 27 и рисунке 28 представлены отображение отзывов и адресов

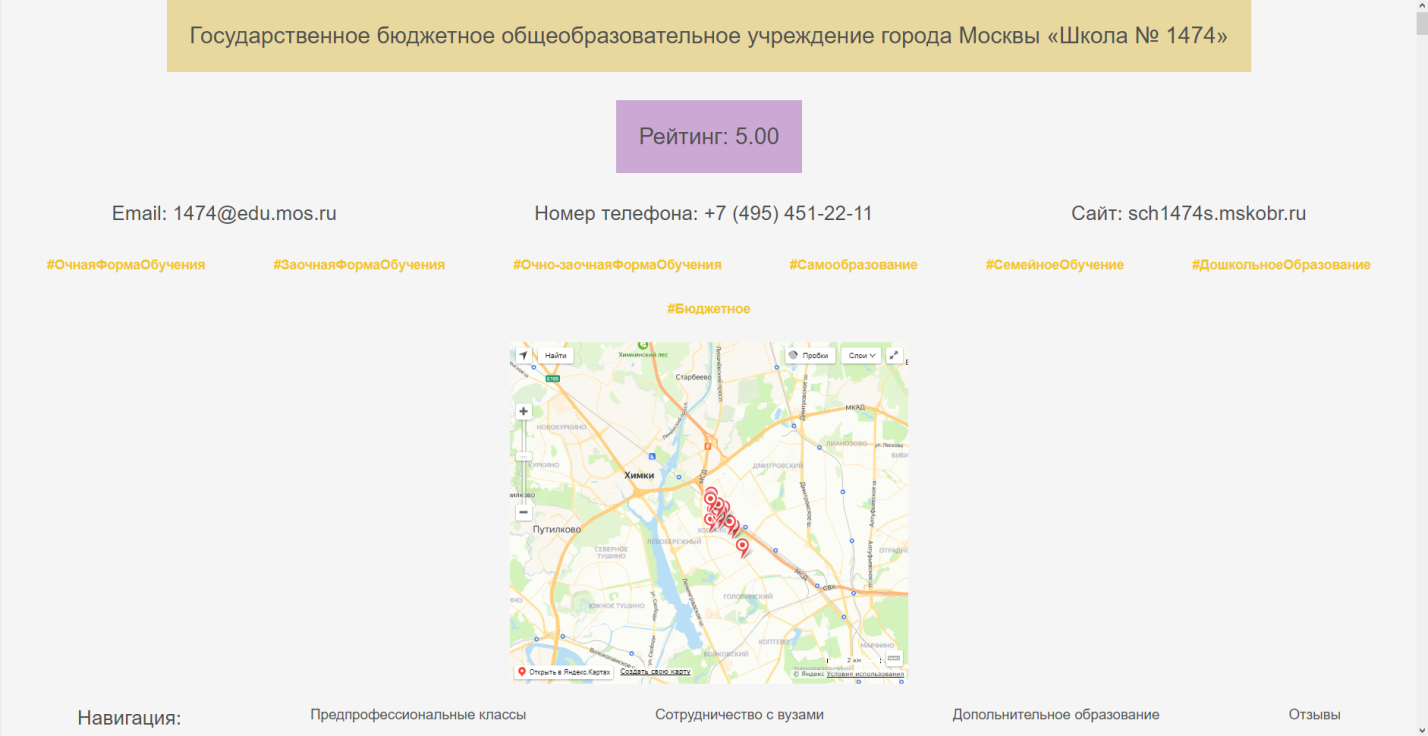


Рисунок 26 - Страница школы

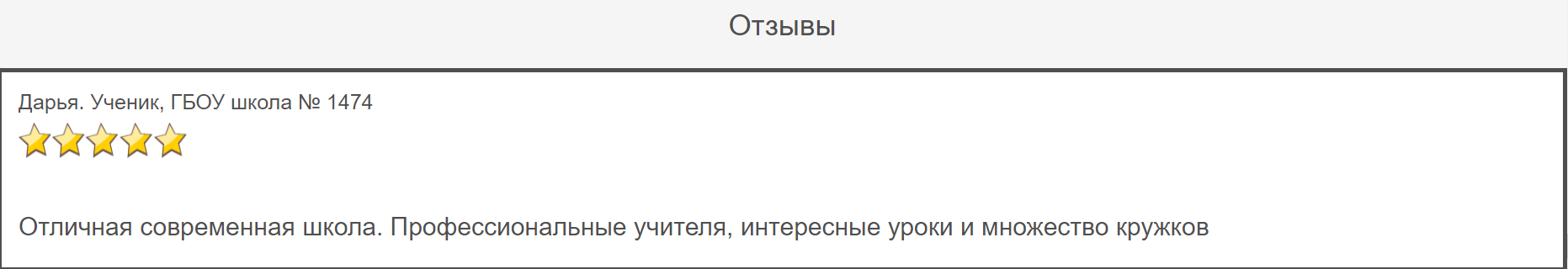


Рисунок 27 - Отображение отзыва

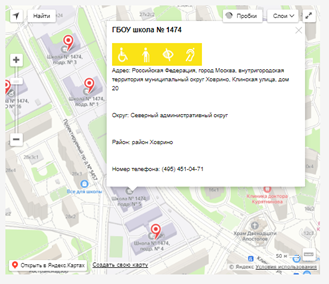


Рисунок 28 - Отображение адресов

### 3.4.4 Шаблон профиля profile.html

В этом шаблоне реализованы два select поля (поля выбора из множества). В них загружаются уже существующие данные или устанавливаются данные по умолчанию.

При изменении роли пользователя изменяется также текст перед полем школы. Функция, в которой это реализовано, представлена на рисунке 29.

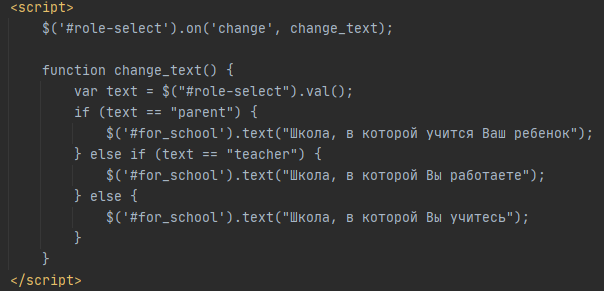


Рисунок 29 - Функция изменения текста на странице профиля

Отображение профиля – рисунок 30.

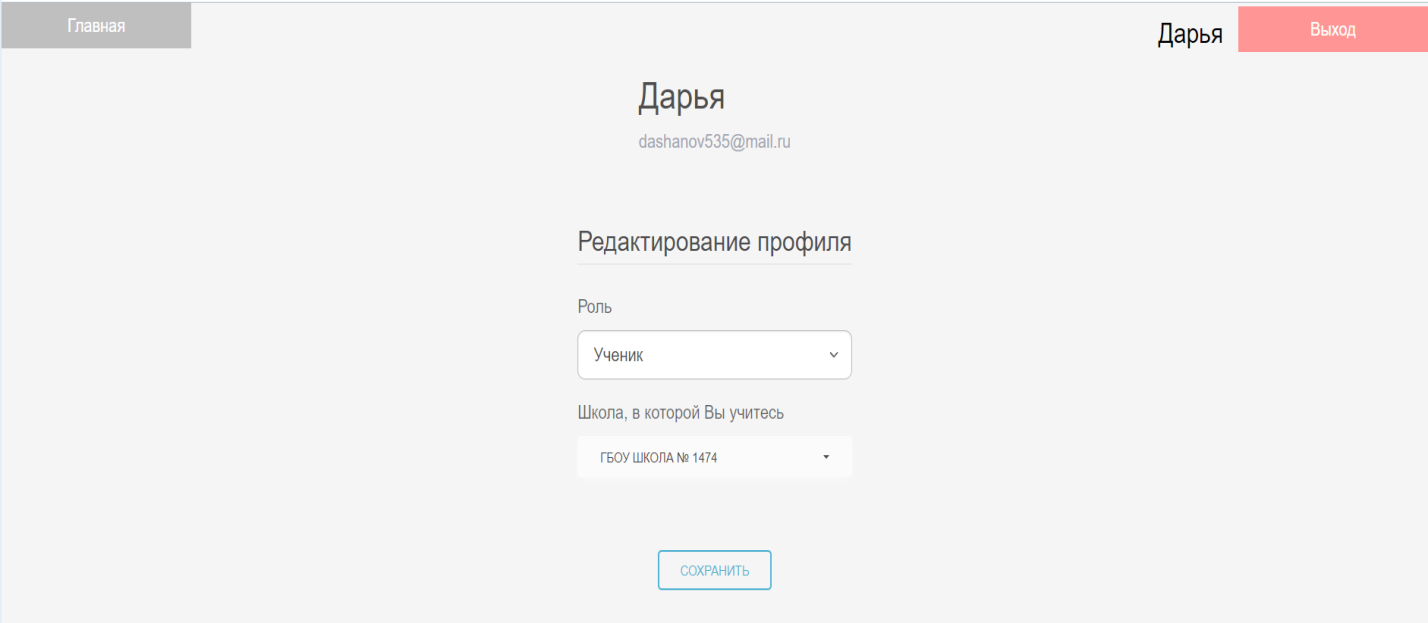


Рисунок 30 – Отображение профиля

### 3.4.5 Шаблоны login.html, register.html и add\_review.html

Шаблоны страниц авторизации, регистрации и добавления отзыва очень похожи. В них отображается переданная обработчиком форма. На рисунке 31 представлен, например, шаблон страницы добавления отзыва.

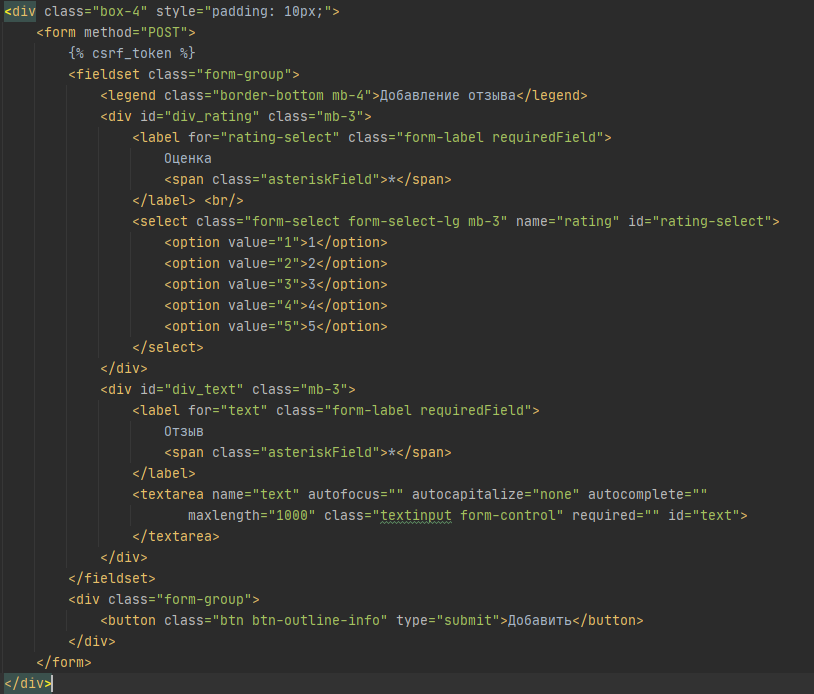


Рисунок 31 - Шаблон добавления отзыва

Отображение страницы добавления отзывов представлено на рисунке 32.



Рисунок 32 - Отображение страницы добавления отзывов

# Заключение

Таким образом, в ходе работы был создан парсер для выгрузки данных о школах с официального сайта и с сайтов каждой школы.

Также была достигнута главная цель проекта – создание сайта. Наш сайт содержит структурированную информацию о школах Москвы и позволяет фильтровать ее по различным критериям. На сайте можно посмотреть предпрофильные классы, секции, вузы-партнеры, отзывы и адреса всех школ Москвы.

Наш проект может помочь быстрее найти необходимую информацию о школе и на ее основе сделать выбор учебного заведения. Перспективой проекта является динамическое обновление данных на сайте.

# Список использованной литературы

1. Портал открытых данных правительства Москвы : [сайт]. – 2004 – . – URL: https://data.mos.ru (дата обращения: 02.03.2023). – Текст : электронный
2. Департамент образования и науки города Москвы : [сайт]. – 2016 – . – URL: https://profil.mos.ru (дата обращения: 02.03.2023). – Текст : электронный
3. AwesomeAndrew : [сайт]. – 2019 – . – URL: https://www.awesomeandrew.ru/2019/07/23/начинаем-работать-с-selenium-в-python/#i-3 (дата обращения: 02.03.2023). – Текст : электронный
4. Меле А. Django 2 в примерах  / пер. с  анг. Д.  В.  Плотниковой. – М.: ДМК Пресс, 2019. – 408 с.: ил. ISBN 978-5-97060-746-6
5. Хабр : [сайт]. – 2003 – . – URL: https://habr.com/ru/post/658139/ (дата обращения: 02.03.2023). – Текст : электронный

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)