|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**  **«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**  **(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»** | | | | |
| **Журнал практики** | | | | |
| Институт № 3 | «Системы управления, информатика и электроэнергетика» | | | |
|  |  | | | |
| Кафедра | №304 | | Учебная группа | М3О-211Б-22 |
|  |  | |  |  |
| ФИО обучающегося | | Мороз Анастасия Ивановна | | |
|  | |  | | |
| Направление подготовки/  специальность | | 09.03.04        Программная инженерия | | |
|  | | *шифр, наименование направления подготовки/специальности* | | |
|  | |  | | |
| Вид практики | | Производственная | | |
|  | | *учебная, производственная, преддипломная или другой вид практики* | | |
| Оценка за практику | | Ведьманов И.С. | | |

Москва

2025

|  |  |  |
| --- | --- | --- |
| 1. **Место и сроки проведения практики:** | | |
| Наименование организации: | ТФОМС Республики Татарстан | |
| Сроки проведения практики |  | |
| дата начала практики: | 30.06.2025 | |
| дата окончания практики: | 27.07.2025 | |
| 1. **Инструктаж по технике безопасности:** | | |
|  | / Гордеев Д.А. / | 30 июня 2025г. |
| *подпись проводившего* | *расшифровка подписи* | *дата проведения* |
| 1. **Индивидуальное задание обучающегося:** | | |
| Индивидуальные задачи обучающегося - загрузка и обработка Excel-файлов, геокодирование и отображение адресов, обработка возможных ошибок, рисование полигонов, тестирование и разработка инструкции по запуску проекта.  Цель - освоить основы работы с Яндекс Геокодером и JS API, разработав сервис, позволяющий наглядно отображать данные о районах обслуживания поликлиник. | | |

|  |
| --- |
| 1. **План выполнения индивидуального задания обучающегося:** |

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Место**  **проведения** | **Тема** | **Период выполнения** |
| 1 | ТФОМС  Республики  Татарстан | Инструктаж. | 30.06.2025 |
| 2 | ТФОМС  Республики  Татарстан | Знакомство с отделом. Получение задания на время практики. | 01.07.2025 |
| 3 | ТФОМС  Республики  Татарстан | Реализация загрузки Excel-файлов, настройка шаблона upload.html для отображения формы загрузки. | 07.07.2025 |
| 4 | ТФОМС  Республики  Татарстан | Обработка геокодирования адресов в файле, отображение адресов, вывод ошибок. | 10.07.2025 |
| 5 | ТФОМС  Республики  Татарстан | Реализация рисования полигонов, настройка стилей рисования. | 15.07.2025 |
| 6 | ТФОМС  Республики  Татарстан | Тестирование загрузки Excel-файлов с разными данными, проверка правильности отображения, тестирование рисования полигонов. | 17.07.2025 |
| 7 | ТФОМС  Республики  Татарстан | Подготовка инструкций по запуску проекта. | 20.07.2025 |
| 8 | ТФОМС  Республики  Татарстан | Составление индивидуального отчета о выполненной работе. | 22.07.2025 |
| 9 | ТФОМС  Республики  Татарстан | Сдача отчёта. | 25.07.2025 |

|  |  |  |
| --- | --- | --- |
| **Утверждаю** | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | / Ведьманов И.С. / | 30 июня 2025г. |
| *подпись руководителя от МАИ* | *расшифровка подписи* | *дата утверждения\** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | / Гордеев Д.А. / | 30 июня 2025г. |
| *подпись руководителя от организации/предприятия* | *расшифровка подписи* | *дата утверждения\** |
| **Ознакомлен** | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | / Мороз А.И. / | 30 июня 2025г. |
| *подпись обучающегося* | *расшифровка подписи* | *дата ознакомления\** |

|  |  |  |
| --- | --- | --- |
| 1. **Отзыв руководителя практики от организации/предприятия:** | | |
| Студентка группы М3О-311Б-22 Мороз Анастасия Ивановна в период с 30.06.2024 по 25.07.2024 проходил производственную практику на предприятии ТФОМС Республики Татарстан (Государственное учреждение «Территориальный фонд обязательного медицинского страхования Республики Татарстан»).  В рамках индивидуального задания Мороз А.И. разработала интерактивный веб-сервис, предназначенный для отображения данных о районах обслуживания поликлиник. Входные данные загружались из файла формата MS Excel, после чего с использованием Яндекс Геокодера осуществлялось преобразование адресов в координаты, а средствами JavaScript API данные отображались на карте с визуальным выделением соответствующих районов.  В процессе выполнения задания она уверенно освоила работу с API-интерфейсами, научилась использовать асинхронные запросы, обрабатывать геоданные и интегрировать их с интерфейсом карты. Также продемонстрировала умение систематизировать информацию, настраивать отображение пользовательского интерфейса и обеспечивать корректную обработку входных данных.  Индивидуальное задание выполнено в полном объёме, с соблюдением всех требований. Представленные в отчёте материалы подтверждают высокий уровень освоения заявленных технологий и способность применять их для решения практических задач.  По итогам практики заслуживает оценку «отлично». | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | / Гордеев Д.А. / | 25 июля 2025г. |
| *подпись руководителя от организации/предприятии* | *расшифровка подписи* | *дата* |

|  |
| --- |
| 1. **Отчет обучающего по практике:** |

Содержание

[О предприятии 6](#_Toc204337943)

[Цели и задачи 7](#_Toc204337944)

[Индивидуальное задание 7](#_Toc204337945)

[Теория 8](#_Toc204337946)

[1. Геокодирование и Яндекс Геокодер 8](#_Toc204337947)

[2. JavaScript API от Яндекс: отображение информации на карте 8](#_Toc204337948)

[3. Сравнение координат с контрольными значениями 8](#_Toc204337949)

[4. Визуализация зон обслуживания 9](#_Toc204337950)

[Описание программного обеспечения 9](#_Toc204337951)

[Структура программы 9](#_Toc204337952)

[Программный код 10](#_Toc204337953)

[Инструкция по запуску и использованию веб-сервиса 18](#_Toc204337954)

[1. Требования к системе 18](#_Toc204337955)

[2. Установка зависимостей 18](#_Toc204337956)

[3. Настройка проекта 18](#_Toc204337957)

[4. Запуск сервера 19](#_Toc204337958)

[5. Использование приложения 19](#_Toc204337959)

[Заключение 22](#_Toc204337960)

# **О предприятии**

**Территориальный фонд обязательного медицинского страхования Республики Татарстан** является ключевым институтом в системе финансирования здравоохранения региона и играет центральную роль в обеспечении реализации государственной политики в сфере обязательного медицинского страхования. Основанный в соответствии с федеральным законодательством, ТФОМС РТ на протяжении многих лет осуществляет стабильную и эффективную деятельность, направленную на гарантированное обеспечение прав граждан на получение бесплатной медицинской помощи в рамках программы ОМС.

Фонд активно внедряет современные цифровые технологии, автоматизированные информационные системы и аналитические платформы для повышения прозрачности, качества и доступности медицинской помощи. ТФОМС РТ выступает организатором и координатором взаимодействия между медицинскими организациями, страховыми компаниями и застрахованными лицами, обеспечивая высокий уровень контроля качества предоставляемых услуг и рациональное использование средств ОМС.

Благодаря интеграции современных решений в области информационных технологий и анализа больших данных, Фонд обеспечивает своевременное принятие управленческих решений, направленных на оптимизацию маршрутизации пациентов, профилактику заболеваний, снижение нагрузки на систему здравоохранения и повышение эффективности расходов.

Основные направления деятельности ТФОМС Республики Татарстан включают:

* организация финансового обеспечения территориальной программы ОМС;
* контроль за объемами, сроками, качеством и условиями предоставления медицинской помощи в системе ОМС;
* анализ и прогнозирование потребностей в медицинских услугах с использованием современных аналитических инструментов;
* сопровождение и развитие единой информационной системы ОМС, интеграция с федеральными цифровыми платформами;
* взаимодействие с медицинскими организациями по вопросам тарификации и контроля за исполнением договоров;
* реализация мероприятий по обеспечению прав застрахованных лиц и защите их интересов;
* мониторинг эффективности использования средств ОМС и подготовка отчетности для органов государственной власти;
* участие в разработке нормативно-правовой базы в сфере обязательного медицинского страхования.

# **Цели и задачи**

При прохождении практики были поставлены следующие цели и задачи:

* изучить направления деятельности Территориального фонда обязательного медицинского страхования Республики Татарстан, включая механизмы сбора, обработки и представления данных о медицинских учреждениях и районах их обслуживания;
* изучить возможности Яндекс.Геокодера для преобразования адресов в географические координаты;
* освоить работу с JavaScript API Яндекс.Карт для интерактивного отображения данных;
* научиться применять инструменты для рисования полигонов и линий на карте;
* на основе полученных знаний и практических навыков реализовать веб-сервис, который считывает данные из Excel-файла и визуализирует районы обслуживания поликлиник на карте города с помощью инструментов Яндекс Геокодера и JS API.

# **Индивидуальное задание**

1. Реализация загрузки Excel-файлов, настройка шаблона upload.html для отображения формы загрузки;
2. Обработка геокодирования адресов в файле, отображение адресов, вывод ошибок для ненайденных адресов или тех, которые указаны некорректно:

* в одном из полей написано «ВСЕ»
* в одном из полей пусто
* в одном из полей введено неправильное написание адреса;

1. Реализация рисования полигонов, настройка стилей рисования;
2. Тестирование загрузки Excel-файлов с разными данными, проверка правильности отображения, тестирование рисования полигонов;
3. Подготовка инструкций по запуску проекта;
4. Составление индивидуального отчета о выполненной работе.

# **Теория**

### **1. Геокодирование и Яндекс Геокодер**

**Геокодирование** - процесс преобразования адреса в географические координаты (широту и долготу). Обратный процесс - определение адреса по координатам - называется **обратным геокодированием**.

В данном проекте используется **Яндекс Геокодер** - часть картографического инструментария Яндекса, доступная через REST API или JavaScript API. Геокодер позволяет:

* получать координаты по текстовому адресу;
* задавать региональные ограничения (например, ограничить поиск только рамками Республики Татарстан);
* работать с результатами в формате JSON.

Листинг 1 - Пример запроса:

|  |
| --- |
| ymaps.geocode("Казань, ул. Вишневского, 1").then(function (res) {  var coords = res.geoObjects.get(0).geometry.getCoordinates();  console.log(coords);  }); |

Чтобы корректно использовать геокодер, необходимо подключить Яндекс.Карты API с полученным ключом и инициализировать карту.

### **2. JavaScript API от Яндекс: отображение информации на карте**

**JavaScript API от Яндекс.Карт** предоставляет обширные возможности для визуализации пространственных данных. Он включает следующие компоненты:

* Базовая карта (спутниковая, гибридная, схема);
* Геообъекты: маркеры, линии, полигоны;
* Работа с событиями и кастомизация элементов;
* Кластеризация объектов на карте;
* Поддержка масштабирования и интерактивности.

В данном задании реализуется отображение:

* Точек (поликлиники) с помощью меток (Placemark);

### **3. Сравнение координат с контрольными значениями**

Важной частью реализации является проверка корректности полученных координат. Сравнение может проводиться:

* по визуальному соответствию - адрес должен располагаться в ожидаемой части карты;
* по точности координат - координаты, полученные из геокодера, должны укладываться в заданный допуск по сравнению с контрольными;
* путём проверки попадания геокоординаты в ожидаемый полигон (район).

### **4. Визуализация зон обслуживания**

Зоны обслуживания могут отображаться в виде:

* полигонов, если известны границы районов;
* групп точек, объединённых по общему признаку.

Использование разных цветов и чекбоксов позволяет пользователю включать и выключать отображение данных по отдельным районам или поликлиникам.

# **Описание программного обеспечения**

Написание программного кода на языке Python выполнено в IDE PyCharm с использованием фреймворка Django.

Результаты работы программы доступны при переходе на веб-страницу, за обработку которой отвечает веб-сервер разрабатываемого сервиса.

# **Структура программы**

Основные файлы Django-проекта (TatOMS):

* **settings.py** - конфигурация проекта
* **urls.py** - маршрутизация URL (главная страница, админ-панель)
* **wsgi.py, asgi.py** - настройки для запуска сервера
* **manage.py** - управление проектом

Приложение (app):

* **views.py** - логика обработки загружаемых Excel-файлов, парсинг данных, передача в шаблон
* **apps.py** - конфигурация приложения

Шаблоны и статика:

* **upload.html** - главная страница: форма загрузки Excel, контейнер для карты, скрипты для работы с API

Вспомогательные файлы:

* **.env** - хранит секретные данные (API-ключ Яндекс.Карт) - YANDEX\_MAPS\_API\_KEY=21353f1f-d598-4e20-a4c8-216ead764e81

# **Программный код**

Полный проект: <https://github.com/NastyushkaM/OMS_practic>

**Выполнение индивидуального задания**

1. **Реализация загрузки Excel-файлов, настройка шаблона upload.html для отображения формы загрузки:**

**views.py:**

*1) Обработка загрузки Excel-файла*

def upload\_file(request):

"""Обрабатывает загруженный Excel файл с адресами"""

addresses = [] # Список валидных адресов

filename = None # Имя загруженного файла

if request.method == "POST" and request.FILES.get('excel\_file'):

excel\_file = request.FILES['excel\_file']

filename = excel\_file.name

# Загрузка Excel файла

wb = openpyxl.load\_workbook(excel\_file)

sheet = wb.active

*2) Этот фрагмент кода возвращает HTML-страницу (upload.html) с переданными данными*

return render(request, 'upload.html', {

'addresses': addresses,

'filename': filename,

'yandex\_api\_key': settings.YANDEX\_MAPS\_API\_KEY,

**upload.html:**

*1)*

/\* Стили формы загрузки \*/

form {

margin-bottom: 10px; //отступ снизу

max-width: 1000px; // максимальная ширина

}

*2) HTML-форма для загрузки Excel-файлов*

<!-- Форма загрузки файла -->

<h2>Загрузка Excel файла</h2>

<form method="post" enctype="multipart/form-data">

{% csrf\_token %} // тег защиты от CSRF-атак

<input type="file" name="excel\_file" accept=".xlsx" required><br><br>

<button type="submit">Загрузить</button>

</form>

<!-- Информация о загруженном файле -->

{% if filename %}

<p>Загружен файл: <strong>{{ filename }}</strong></p>

{% endif %}

1. **Обработка геокодирования адресов в файле, отображение адресов, вывод ошибок для ненайденных адресов или тех, которые указаны некорректно:**

* **в одном из полей написано «ВСЕ»**
* **в одном из полей пусто**
* **в одном из полей введено неправильное написание адреса;**

**views.py:**

*1)*

all\_houses\_rows = [] # Строки с "ВСЕ" в номерах домов

all\_streets\_rows = [] # Строки с "ВСЕ" в названиях улиц

empty\_field\_rows = [] # Строки с пустыми обязательными полями

invalid\_houses\_rows = [] # Строки с невалидными номерами домов

not\_found\_addresses\_rows = [] # Строки с адресами, которые не найдены на карте

*2) Обработка данных из Excel-файла*

# Пропускаем заголовок (первая строка)

for row\_num, row in enumerate(sheet.iter\_rows(min\_row=2, values\_only=True), start=2):

# Проверка на пустые поля

has\_empty\_fields = False

if not row[0]: # Номер участка

has\_empty\_fields = True

if not row[1] or not str(row[1]).strip(): # Город

has\_empty\_fields = True

if not row[2] or not str(row[2]).strip(): # Улица

has\_empty\_fields = True

if not row[3] or not str(row[3]).strip(): # Дома

has\_empty\_fields = True

if has\_empty\_fields:

empty\_field\_rows.append(row\_num)

continue # Пропускаем строку с пустыми полями

# Извлечение данных из строки

group = int(row[0]) if row[0] else None

city = row[1].strip() if row[1] else ''

street = row[2].strip() if row[2] else ''

houses\_str = str(row[3]) if row[3] else ''

# Проверяем поле улицы на "ВСЕ"

if street.upper() == "ВСЕ":

all\_streets\_rows.append(row\_num)

continue # Пропускаем строку с "ВСЕ" в поле улицы

# Проверяем на невалидные значения

if "INVALID" in houses:

invalid\_houses\_rows.append(row\_num)

continue

# Проверяем поле списка домов на "ВСЕ"

if "ВСЕ" in houses:

all\_houses\_rows.append(row\_num)

continue

# Обработка специального случая

if "Лесной городок" in houses:

addresses.append({

'group': group,

'city': city,

'street': street,

'house': 'Лесной городок',

'is\_forest\_town': True,

'row\_num': row\_num

})

continue

# Добавление всех домов из строки

if houses:

for house in houses:

addresses.append({

'group': group,

'city': city,

'street': street,

'house': house,

'row\_num': row\_num

})

else:

# Если домов нет, добавляем только улицу

addresses.append({

'group': group,

'city': city,

'street': street,

'house': None

})

# Формирование сообщений об ошибках

# Сообщение о строках с "ВСЕ" в улицах

if all\_streets\_rows:

if len(all\_streets\_rows) == 1:

row\_word = "строку"

err\_street\_word = "названия улицы"

else:

row\_word = "строки"

err\_street\_word = "названия улицы"

messages.error(

request,

f'Обнаружена ошибка! Вместо {err\_street\_word} написано "ВСЕ". Эти адреса не были обработаны. '

f'Исправьте, пожалуйста, {row\_word}:'

)

# Сообщение о строках с "ВСЕ" в домах

if all\_houses\_rows:

if len(all\_houses\_rows) == 1:

row\_word = "строку"

err\_houses\_word = "номеров домов"

else:

row\_word = "строки"

err\_houses\_word = "номеров домов"

messages.error(

request,

f'Обнаружена ошибка! Вместо {err\_houses\_word} написано "ВСЕ". Эти адреса не были обработаны. '

f'Исправьте, пожалуйста, {row\_word}:'

)

# Сообщение о строках с пустыми полями

if empty\_field\_rows:

if len(empty\_field\_rows) == 1:

row\_word = "строку"

field\_word = "пустыми полями"

else:

row\_word = "строки"

field\_word = "пустыми полями"

messages.error(

request,

f'Обнаружена ошибка! Есть строки с {field\_word}. Эти адреса не были обработаны. '

f'Исправьте, пожалуйста, {row\_word}:'

)

# Сообщение о невалидных номерах домов

if invalid\_houses\_rows:

if len(invalid\_houses\_rows) == 1:

row\_word = "строку"

error\_word = "некорректные значения"

else:

row\_word = "строки"

error\_word = "некорректные значения"

messages.error(

request,

f'Обнаружена ошибка! В файле указаны {error\_word}. Эти адреса не были обработаны. '

f'Исправьте, пожалуйста, {row\_word}:'

)

*3)*

'all\_houses\_rows': all\_houses\_rows,

'all\_streets\_rows': all\_streets\_rows,

'empty\_field\_rows': empty\_field\_rows,

'invalid\_houses\_rows': invalid\_houses\_rows,

'not\_found\_addresses\_rows': not\_found\_addresses\_rows,

})

**upload.html:**

*1)*

/\* Стили сообщений \*/

.alert {

padding: 10px 15px; /\* Внутренние отступы: 10px сверху/снизу, 15px слева/справа \*/

margin-bottom: 15px; /\* Внешний отступ снизу 15px \*/

border: 1px solid transparent; /\* Прозрачная граница толщиной 1px \*/

border-radius: 4px; /\* Закругление углов радиусом 4px \*/

}

.alert-error {

color: #a94442; /\* Темно-красный цвет текста \*/

background-color: #f2dede; /\* Светло-красный фон \*/

border-color: #ebccd1; /\* Розовато-красная граница \*/

}

.toggle-errors {

color: #5d8aa8; /\* Синий цвет текста (как ссылка) \*/

cursor: pointer; /\* Меняет курсор на "указатель" при наведении \*/

}

.alert-warning {

color: #856404; /\* Темно-желтый/коричневый цвет текста \*

background-color: #fff3cd; /\* Светло-желтый фон \*/

border-color: #ffeeba; /\* Желтая граница \*/

}

*2) Этот фрагмент кода отвечает за отображение и управление системой уведомлений об ошибках*

<!-- Сообщения об ошибках -->

{% if messages %}

<div class="messages">

{% for message in messages %}

<div class="alert alert-{{ message.tags }}">

{{ message }}

{% if all\_houses\_rows and "номеров домов" in message|stringformat:"s" or all\_streets\_rows and "названия улицы" in message|stringformat:"s" or empty\_field\_rows and "пустыми полями" in message|stringformat:"s" or invalid\_houses\_rows and "некорректные значения" in message|stringformat:"s" %}

<div class="error-details" style="margin-top: 4px;">

<a href="#" class="toggle-errors">показать номера строк в файле</a>

<div class="full-error-list" style="display: none; max-height: 150px; overflow-y: auto;">

{% if "номеров домов" in message|stringformat:"s" %}

{{ all\_houses\_rows|join:", " }}

{% elif "названия улицы" in message|stringformat:"s" %}

{{ all\_streets\_rows|join:", " }}

{% elif "пустыми полями" in message|stringformat:"s" %}

{{ empty\_field\_rows|join:", " }}

{% else %}

{{ invalid\_houses\_rows|join:", " }}

{% endif %}

</div>

</div>

{% endif %}

</div>

{% endfor %}

</div>

{% endif %}

<script>

// Обработчик для переключения отображения списка ошибок

document.addEventListener('DOMContentLoaded', function() {

const toggleButtons = document.querySelectorAll('.toggle-errors');

toggleButtons.forEach(btn => {

btn.addEventListener('click', function(e) {

e.preventDefault();

const errorList = this.nextElementSibling;

if (errorList.style.display === 'none') {

errorList.style.display = 'block';

this.textContent = 'скрыть список';

} else {

errorList.style.display = 'none';

this.textContent = 'показать номера строк в файле';

}

});

});

});

</script>

<!-- Контейнер для данных адресов и предупреждений -->

{% if addresses %}

{{ addresses|json\_script:"addr-data" }}

<div id="not-found-addresses-warning" class="alert alert-warning" style="display: none;"></div>

{% endif %}

*3) Этот фрагмент выполняет геокодирование адресов и обработку результатов*

// Обработка данных адресов

const addressData = JSON.parse(document.getElementById('addr-data').textContent); // Получает данные адресов

const notFoundAddresses = []; // Массив для ненайденных адресов

const notFoundRows = new Set(); // Уникальные номера строк с ошибками

// Геокодирование

if (item.is\_forest\_town) {

const placemark = new ymaps.Placemark(FOREST\_TOWN\_COORDS, {

hintContent: "ЖК Лесной городок",

balloonContent: `№ ${item.group}: ЖК Лесной городок, ${item.street}`

...

});

...

} else {

// Геокодирование обычного адреса

const address = item.house

? `${item.city}, ${item.street}, ${item.house}`

: `${item.city}, ${item.street}`;

ymaps.geocode(address, {results: 1}).then(res => {

const geoObject = res.geoObjects.get(0);

if (geoObject) {

// Проверяем точность геокодирования

const precision = geoObject.properties.get('metaDataProperty.GeocoderMetaData.precision');

if (precision === 'exact' || precision === 'number' || precision === 'near' || precision === 'range') {

// Адрес найден

...

} else {

// Адрес найден, но с низкой точностью (например, только город)

notFoundAddresses.push({

group: item.group,

address: address,

message: 'Некорректный адрес'

});

if (item.row\_num) notFoundRows.add(item.row\_num);

}

} else {

// Адрес не найден

notFoundAddresses.push({

group: item.group,

address: address,

message: 'Адрес не найден на карте'

});

if (item.row\_num) notFoundRows.add(item.row\_num);

}

}, function(err) {

// Ошибка геокодирования

notFoundAddresses.push({

group: item.group,

address: address,

message: 'Ошибка при поиске адреса: ' + (err.message || 'неизвестная ошибка')

});

if (item.row\_num) notFoundRows.add(item.row\_num);

});

}

});

// Показать список ненайденных адресов

setTimeout(() => {

if (notFoundAddresses.length > 0) {

const warningContainer = document.getElementById('not-found-addresses-warning');

...

warningContainer.style.display = 'block';

...

});

}

}, 2000);

1. **Реализация рисования полигонов, настройка стилей рисования;**

**upload.html:**

*1)*

/\* Стили кнопок \*/

#draw-polygon, #clear-polygon, #find-houses {

padding: 8px 15px; /\* Внутренние отступы: 8px сверху/снизу, 15px слева/справа \*/

margin-right: 10px; /\* Внешний отступ справа 10px между кнопками \*/

background-color: #4CAF50; /\* Зеленый цвет фона \*/

color: white; /\* Белый цвет текста \*/

border: none; /\* Убираем стандартную границу \*/

border-radius: 4px; /\* Скругляем углы на 4px \*/

cursor: pointer; /\* Курсор в виде указателя при наведении \*/

}

#draw-polygon:hover, #clear-polygon:hover, #find-houses:hover {

background-color: #45a049; /\* Темно-зеленый цвет фона при наведении \*/

}

/\* Стили результатов поиска \*/

#polygon-results {

border: 1px solid #ddd; /\* Тонкая серая рамка \*/

padding: 15px; /\* Внутренние отступы 15px со всех сторон \*/

border-radius: 5px; /\* Скругление углов на 5px \*/

background-color: #f9f9f9; /\* Светло-серый фон \*/

}

#missing-houses-list {

max-height: 300px; /\* Максимальная высота 300px с вертикальным скроллом \*/

overflow-y: auto; /\* Автоматическая прокрутка по вертикали при переполнении \*/

margin-top: 10px; /\* Отступ сверху 10px от предыдущего элемента \*/

}

*2) Этот фрагмент кода реализует функционал работы с полигонами на карте*

<!-- Управление полигоном -->

<div style="margin: 10px 0;">

<button id="draw-polygon">Нарисовать полигон</button>

<button id="clear-polygon" style="display: none;">Очистить полигон</button>

<button id="find-houses" style="display: none;">Найти дома внутри полигона</button>

</div>

<!-- Результаты поиска по полигону -->

<div id="polygon-results" style="display: none; margin-top: 20px;">

<h3>Дома внутри территории:</h3>

<div id="missing-houses-list"></div>

</div>

*3) Этот код реализует функционал рисования полигона на Яндекс.Карте с последующим управлением им*

let polygon = null; // Готовый полигон (после завершения рисования)

let polygonDrawingManager = null; // Менеджер рисования (пока пользователь рисует)

// Управление полигоном

// Кнопка рисования полигона

document.getElementById('draw-polygon').addEventListener('click', function() {

// Удаляем старый полигон, если есть

if (polygon) {

myMap.geoObjects.remove(polygon);

polygon = null;

}

// Отключаем скролл карты

myMap.behaviors.disable('scrollZoom');

// Создаем многоугольник

polygonDrawingManager = new ymaps.Polygon([], {}, {

editorDrawingCursor: "crosshair", // Курсор в виде крестика

editorMaxPoints: 100,

fillColor: '#00FF00',

strokeColor: '#0000FF',

strokeWidth: 2,

opacity: 0.5

});

// Добавляем менеджер на карту

myMap.geoObjects.add(polygonDrawingManager);

// Включаем режим рисования

polygonDrawingManager.editor.startDrawing();

// Кнопки управления

document.getElementById('clear-polygon').style.display = 'inline-block';

document.getElementById('draw-polygon').style.display = 'none';

document.getElementById('find-houses').style.display = 'none';

// Обработчик завершения рисования

polygonDrawingManager.editor.events.add('drawingstop', function() {

polygon = polygonDrawingManager; // Сохраняем готовый полигон

polygonDrawingManager = null; / Удаляем менеджер

document.getElementById('find-houses').style.display = 'inline-block'; // Показываем кнопку поиска

myMap.behaviors.enable('scrollZoom'); // Включаем скролл

});

});

// Кнопка очистки полигона

document.getElementById('clear-polygon').addEventListener('click', function() {

// Удаляем полигон с карты

if (polygon) {

myMap.geoObjects.remove(polygon);

polygon = null;

}

// Если есть активный менеджер рисования, удаляем и его

if (polygonDrawingManager) {

myMap.geoObjects.remove(polygonDrawingManager);

polygonDrawingManager = null;

}

// Возвращаем скролл карты

myMap.behaviors.enable('scrollZoom');

// Возвращаем кнопки в исходное состояние

document.getElementById('draw-polygon').style.display = 'inline-block';

document.getElementById('clear-polygon').style.display = 'none';

document.getElementById('find-houses').style.display = 'none';

// Скрываем результаты поиска домов, если они были показаны

document.getElementById('polygon-results').style.display = 'none';

});

# **Инструкция по запуску и использованию веб-сервиса**

### **1. Требования к системе**

* Python (версия 3.8 или выше)
* Django (версия 3.2 или выше)
* Библиотеки: openpyxl (для работы с Excel)
* API-ключ Яндекс.Карт

(получить можно [здесь](https://developer.tech.yandex.ru/services/) - https://developer.tech.yandex.ru/services)

### **2. Установка зависимостей**

* Создайте виртуальное окружение:

python -m venv venv

source venv/bin/activate # для Linux/Mac

venv\Scripts\activate # для Windows

* Установите Django и openpyxl:

pip install django openpyxl

### **3. Настройка проекта**

* Проверьте, что подключен API-ключ Яндекс.Карт в файле .env

### **4. Запуск сервера**

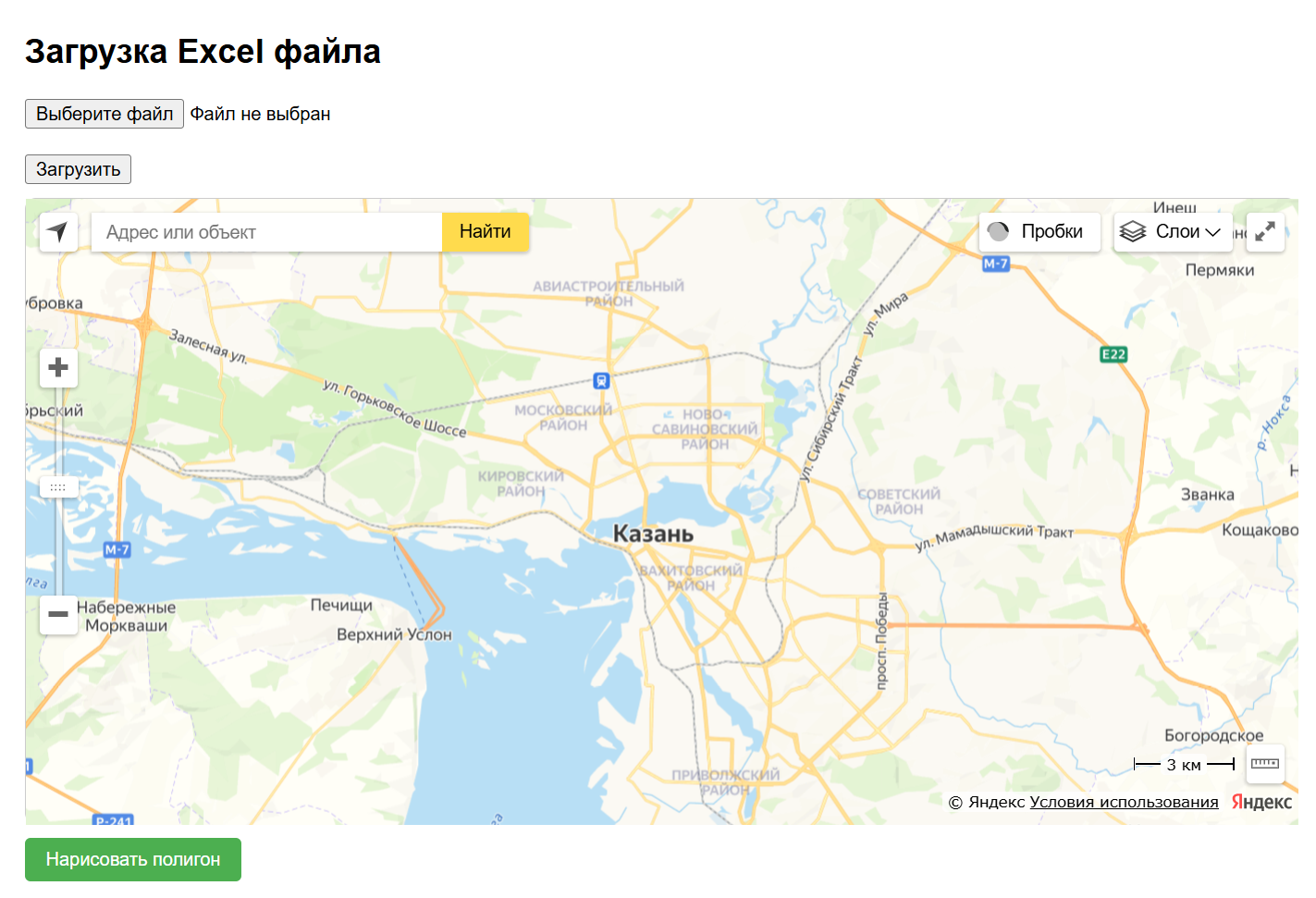
* Выполните команду в терминале:

python manage.py runserver

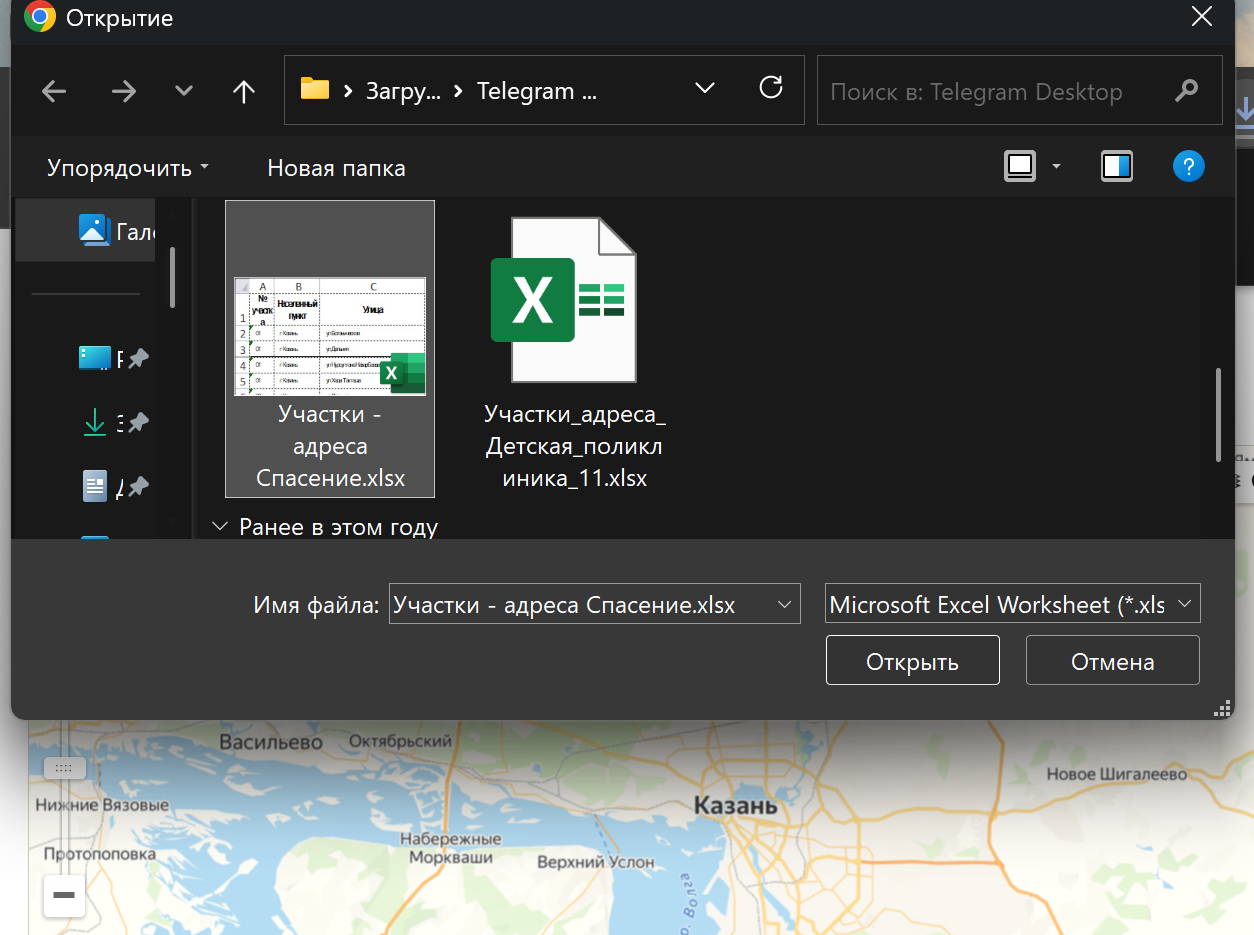
* Запуск сервера - откройте браузер и перейдите по адресу <http://127.0.0.1:8000/> или нажмите на него в терминале, вас автоматически перекинет

### **5. Использование приложения**

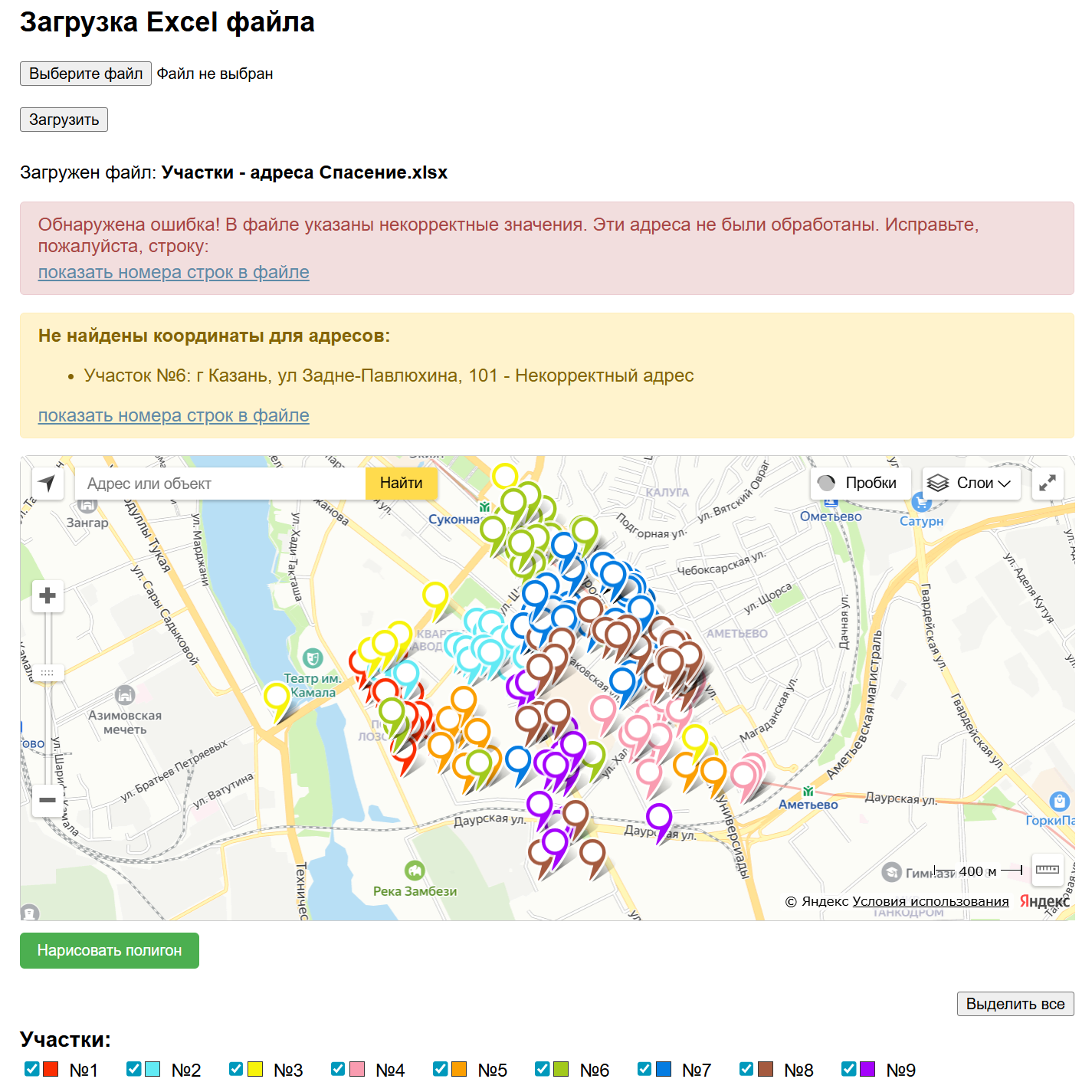
* Когда вы запустите сервер, то увидите следующее:



* Выберите файл, который хотите загрузить:



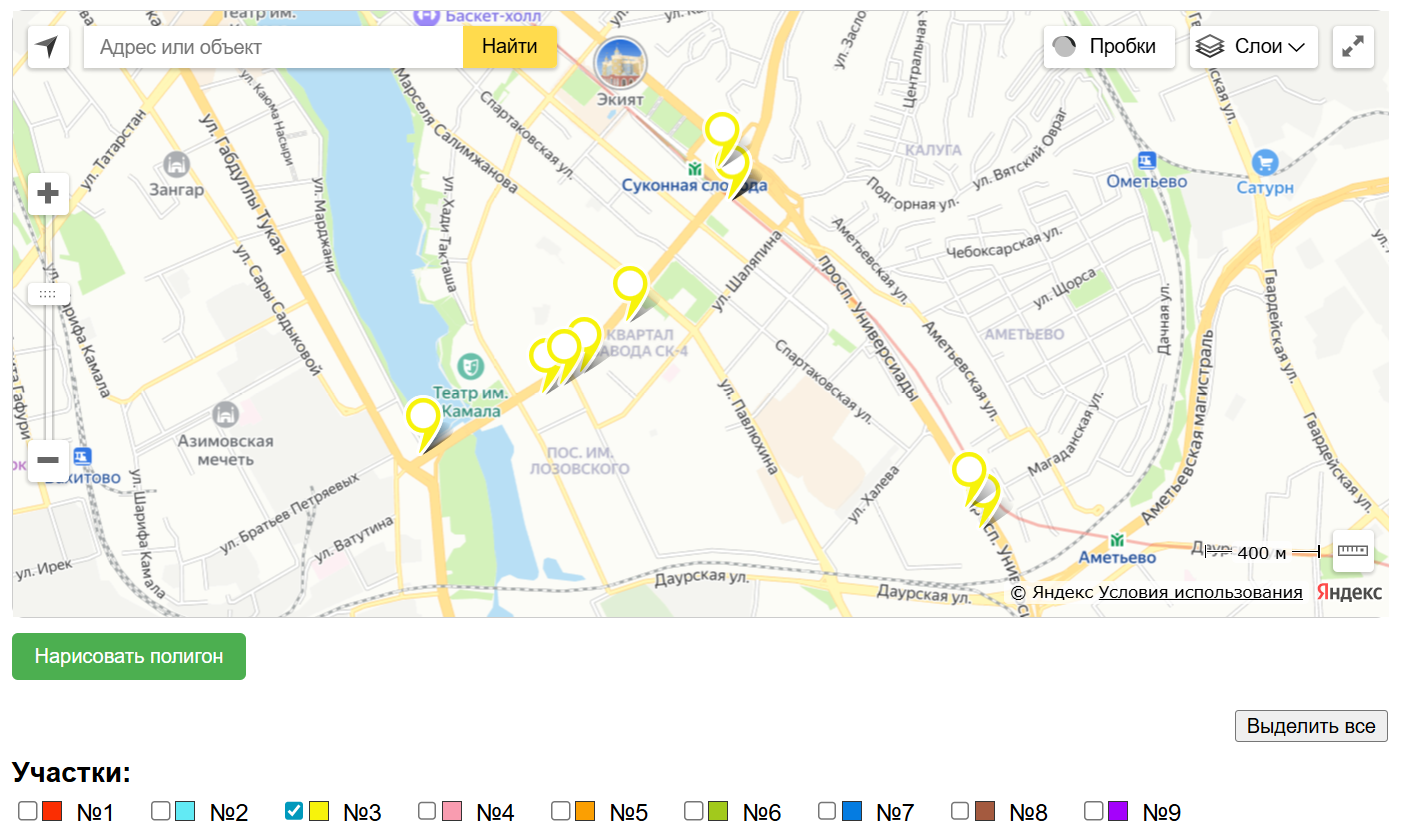
* Загрузите файл:



Для того, чтобы посмотреть строки с ошибками или ненайденными адресами, нажмите «показать номера строк в файле».

Если вы хотите, чтобы все метки пропали, нажмите «Снять все», и наоборот – если хотите, чтобы все метки появились, нажмите «Выделить все».

Если вы хотите, чтобы на карте отображались адреса, которые принадлежат какому-то определенному участку, выберите нужный чекбокс с нужным номером участка.

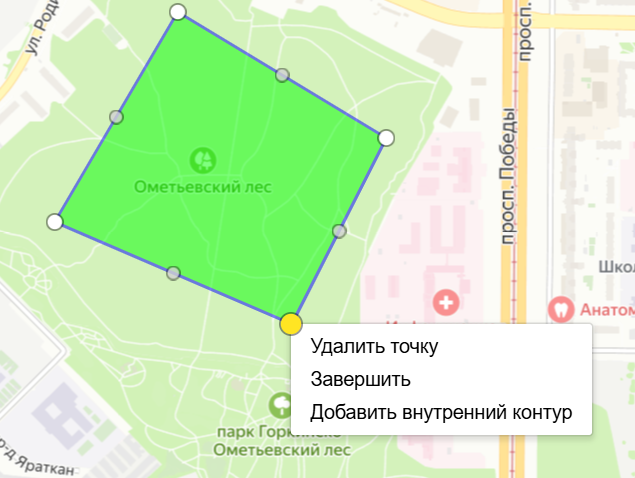


* Работа с полигоном:

1. Нажмите кнопку «Нарисовать полигон»



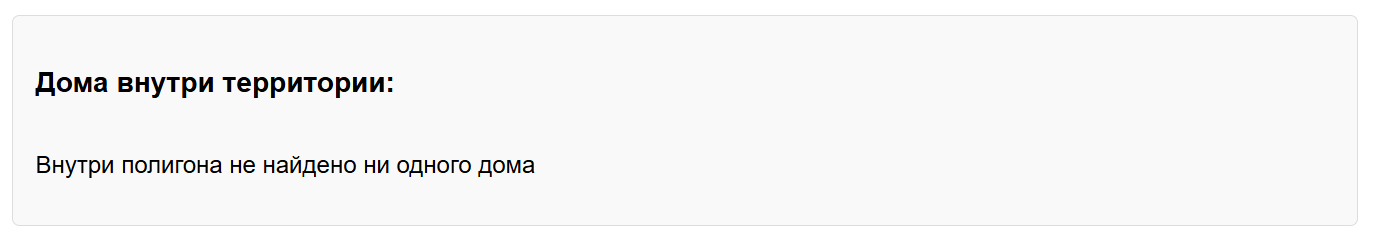
1. Нарисуйте полигон – чтобы завершить отрисовку/удалить точку/продолжить отрисовку нажмите на какую-нибудь точку из тех, которые вы поставили



1. После нажатия в меню позиции «Завершить», появится кнопка «Найти дома внутри полигона», а также поменяется кнопка «Нарисовать полигон» на кнопку «Очистить полигон»



1. Нажмите кнопку «Найти дома внутри полигона», и вы получите информацию о домах внутри выделенного полигона



1. Чтобы сбросить выбранную территорию нажмите «Очистить полигон»

# 

# **Заключение**

В ходе прохождения практики были изучены возможности ЯндексГеокодера и JavaScript API Яндекс.Карт, а также освоены методы работы с библиотекой openpyxl для обработки Excel-файлов.

На основе полученных знаний был разработан веб-сервис, который позволяет загружать данные о районах обслуживания поликлиник, преобразовывать адреса в координаты и отображать их на интерактивной карте с возможностью фильтрации и настройки отображения меток.

Выполнение индивидуальных задач позволило закрепить навыки интеграции внешних API, работы с данными и разработки веб-интерфейсов, что способствовало достижению целей практики.

|  |  |  |
| --- | --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | / Мороз А.И. / | 25 июля 2025г. |
| *подпись обучающегося* | *расшифровка подписи* | *дата* |