Министерство образования Республики Беларусь Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ

И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №4,5

По теме “Разработка защищенных приложений.”

Выполнила: студентка гр. 053501 Шурко Т.А.

Проверил: ассистент кафедры информатики Протько М. И.

Минск 2023

Содержание

[1. Введение 3](file:///D:\downloads\Lab1.docx#_Toc7030)

[2. Теоретические сведения 6](file:///D:\downloads\Lab1.docx#_Toc7032)

[3. Тестирование программного продукта 8](file:///D:\downloads\Lab1.docx#_Toc7033)

[4. Вывод 12](file:///D:\downloads\Lab1.docx#_Toc7035)

Приложение А. Код программы 13

# Введение

Познакомиться с концепцией ролевого управления доступом и способами защиты программного обеспечения от существующих угроз.

Научиться разрабатывать приложения, которые используют ролевое управление доступом для разграничения полномочий пользователей.

Получить навыки защиты разработанной программы от несанкционированного копирования и других угроз, которым может подвергаться программное обеспечение.

# Теоретические сведения

Рассмотрим следующие атаки:

* Атака “переполнение буфера”.

Злоумышленники используют атаки с переполнением буфера, чтобы проникнуть в систему целевого компьютера или повредить хранящиеся там данные.

Во время таких атак переполняется буфер с фиксированным объемом памяти, зарезервированный для процесса ввода. Большой процент атак составляют атаки с переполнением буфера, направленные на перезапись информации в смежных ячейках памяти во фрейме стека.

* Атака “SQL-инъекции”.

Уязвимость SQL инъекции позволяет атакующему выполнить произвольный SQL код в базе данных и получить доступ к данным (прочитать, отредактировать и изменить) независимо от текущих прав доступа пользователя. В большинстве случаев вы будете получать доступ к данным базы данных, используя сущности queryset/model Django. Используя их для генерации SQL запросов, вы получите корректно сформированный и экранированный запрос для выбранной базы данных. Если вам необходимо писать "сырые" запросы, вам так же нужно будет продумать защиту от инъекций.

Фундаментальные правила предотвращения SQL-инъекций:

## **Никогда** не доверяйте никаким данным, предоставленным пользователем;

## **Всегда** используйте "параметризованные операторы" при непосредственном построении SQL-запросов.

* Атака “XSS” (межсайтовое кодирование).

XSS это термин, применяющийся для описания класса атак, позволяющего атакующему, через веб-сайт внедрить скрипты, которые будут выполнены на устройстве зашедшего на страницу пользователя. Часто это происходит через сохранение вредоносного кода в базе данных, откуда данный код будет возвращён и выполнен для запросившего некие данные пользователя (типичный пример - сохранение тега <script> с вредоносным кодом в комментарии, который может увидеть другой пользователь). Другой вектор атаки - в том чтобы сгенерировать определённую ссылку, при клике на которую пользователь запустит выполнение некоего замаскированного кода JavaScript в своём браузере.

Система шаблонов Django защищает от большинства XSS-атак, [экранируя определённые символы](https://docs.djangoproject.com/en/2.0/ref/templates/language/#automatic-html-escaping), считающиеся "опасными" в HTML.

* Принцип минимизации привилегий.

Принцип наименьших привилегий гласит, что модулям программного обеспечения должны предоставляться минимальные возможности, необходимые для выполнения поставленной перед ними задачи.

* Межсайтовая подделка запроса CSRF.

CSRF атаки позволяют атакующему выполнять действия от имени другого пользователя без его согласия.

# Тестирование программного продукта

На рисунках 3.1, 3.2 произведена атака “XSS” и показана реакция программного продукта на атаки такого рода.

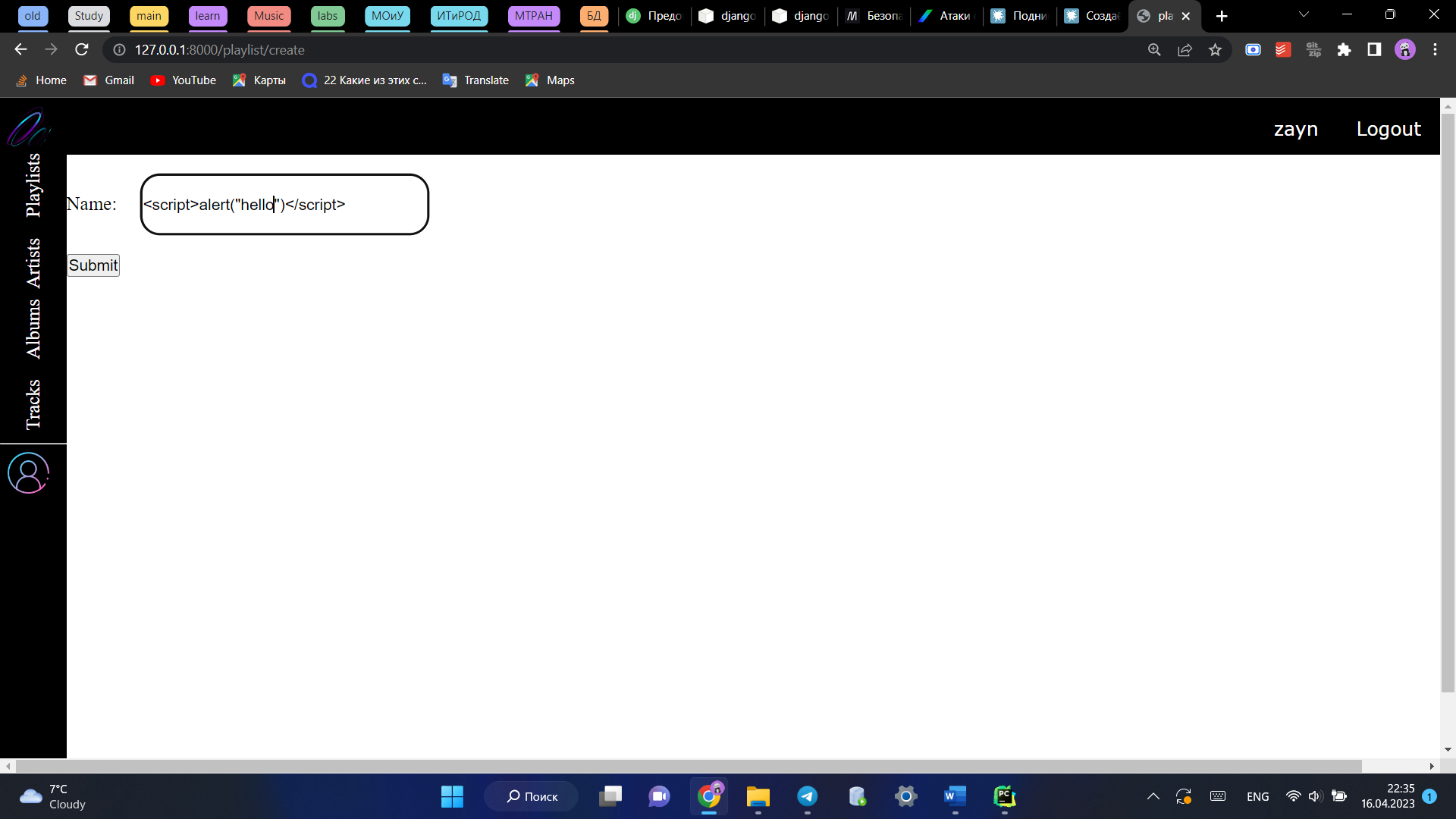


Рисунок 3.1 – Проверка атаки “XSS” (межсайтовое кодирование)

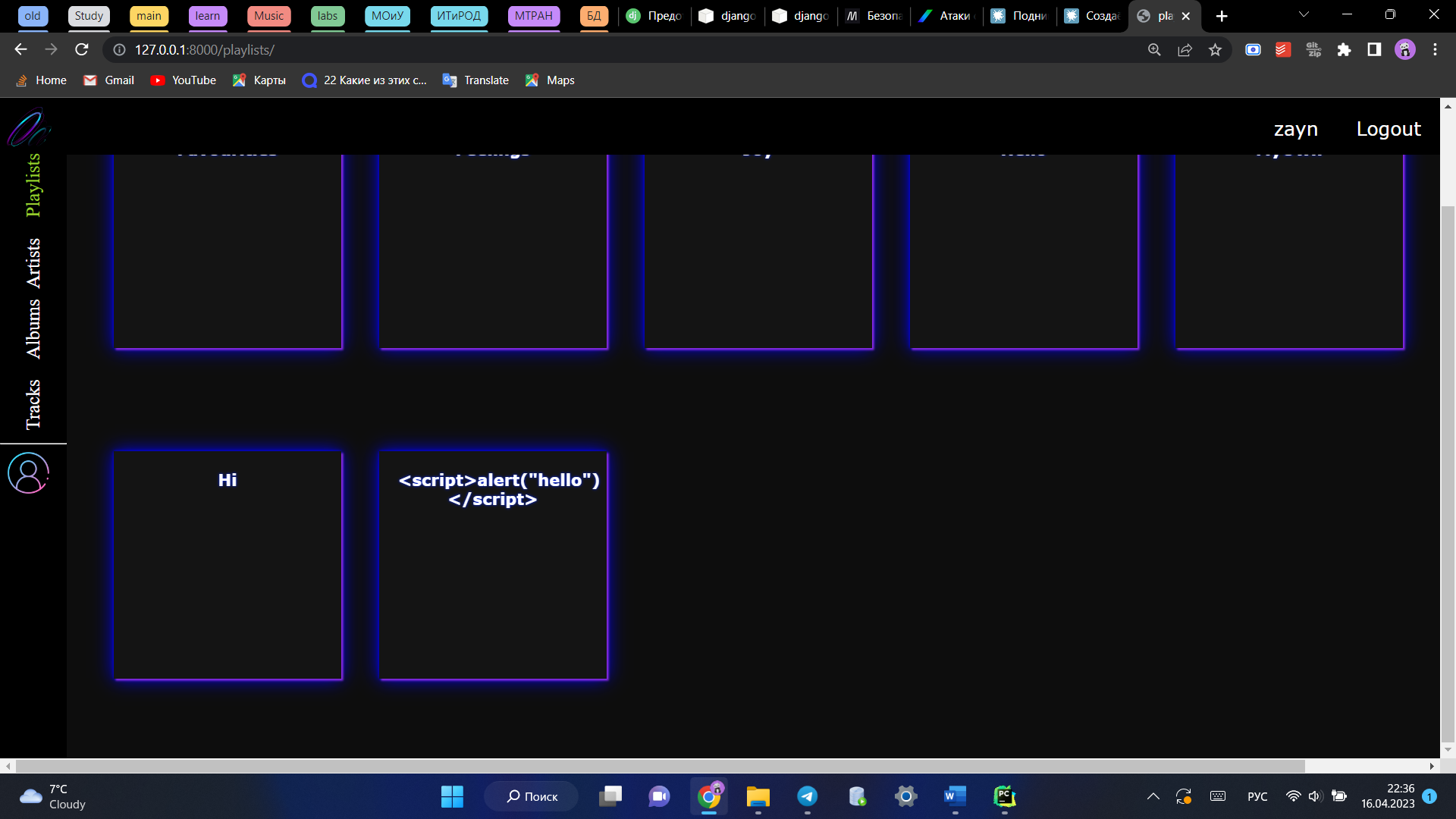


Рисунок 3.2 – Результат атаки “XSS” (межсайтовое кодирование)

На рисунке 3.3 показана модель Playlist, текстовое поле которого ограничено (максимально допустимое число вводимых символов - 150). Таким образом, программа не сможет переполнить буфер, так как все вводимые тестовые поля ограничены сверху.

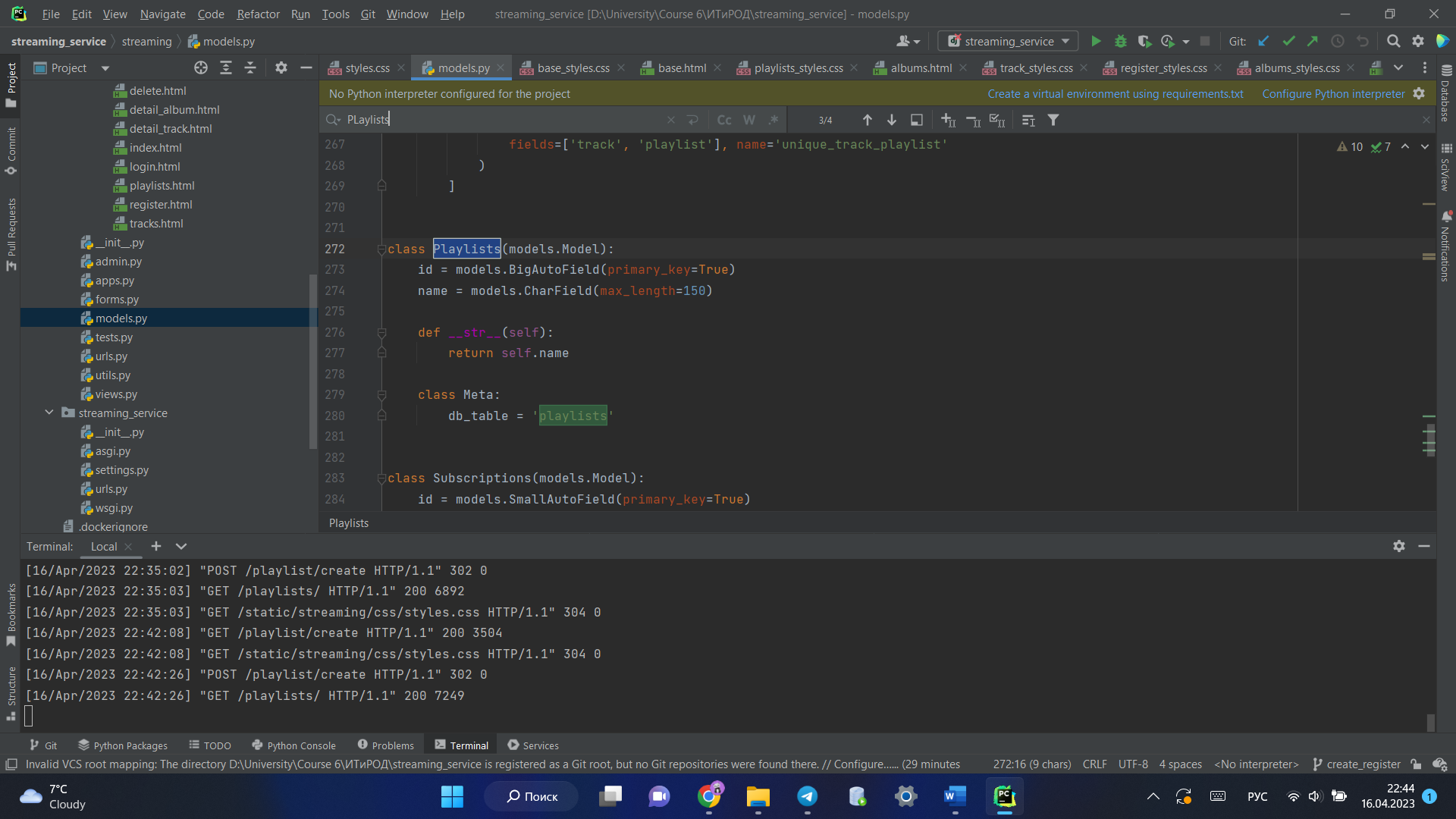


Рисунок 3.3 – Модель Playlist

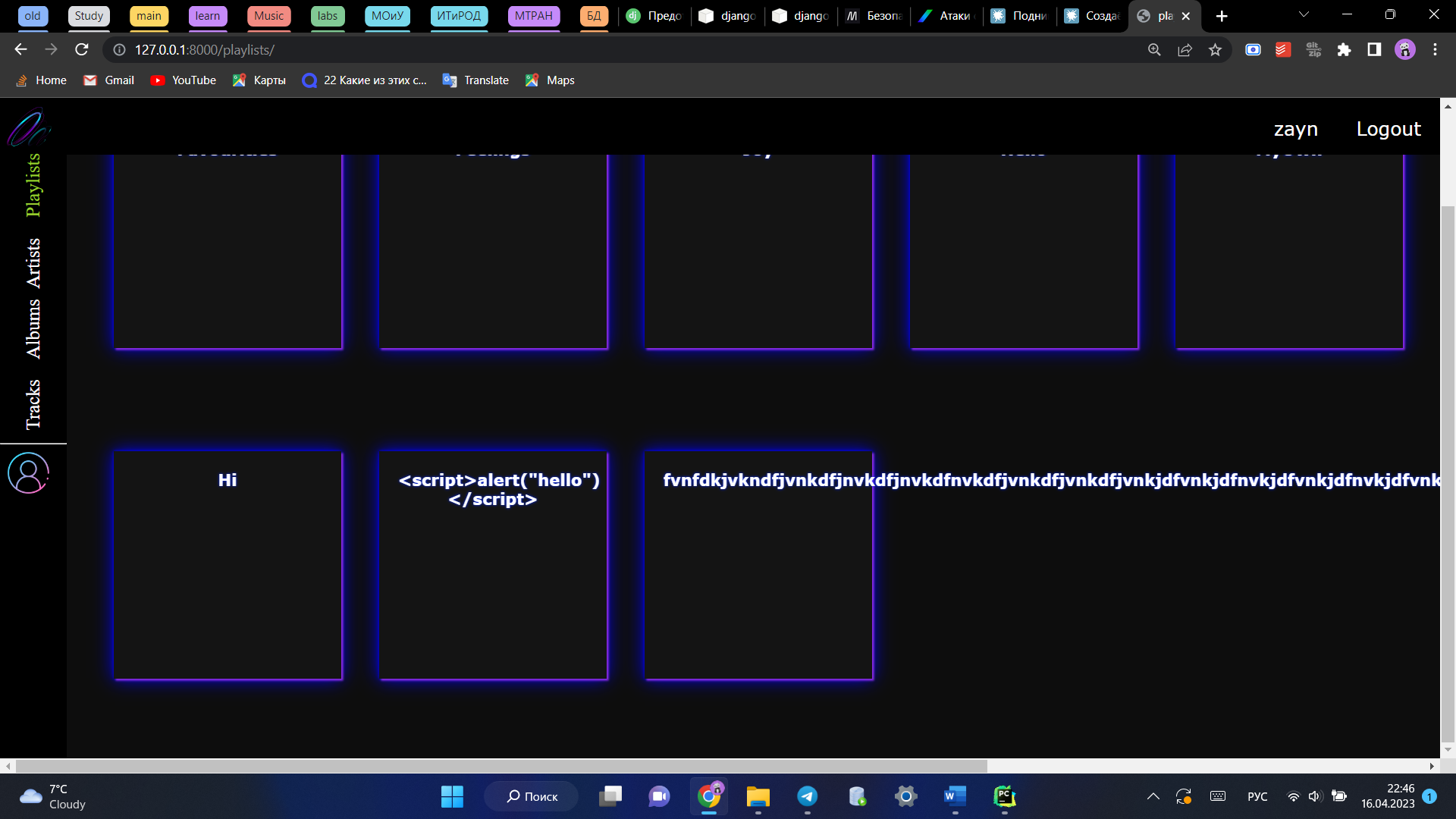


Рисунок 3.4 – Попытка переполнения буфера

Попытка SQL-инъекций также не увенчается успехом, так как запросы обрабатываются с помощью моделей Django. Используя их для генерации SQL запросов, получим корректно сформированный и экранированный запрос для выбранной базы данных.  К тому же, в данном программном продукте в запросах к базе данных не используются напрямую введенные пользователями данные. Пример запроса с помощью ORM на рисунке 3.5.

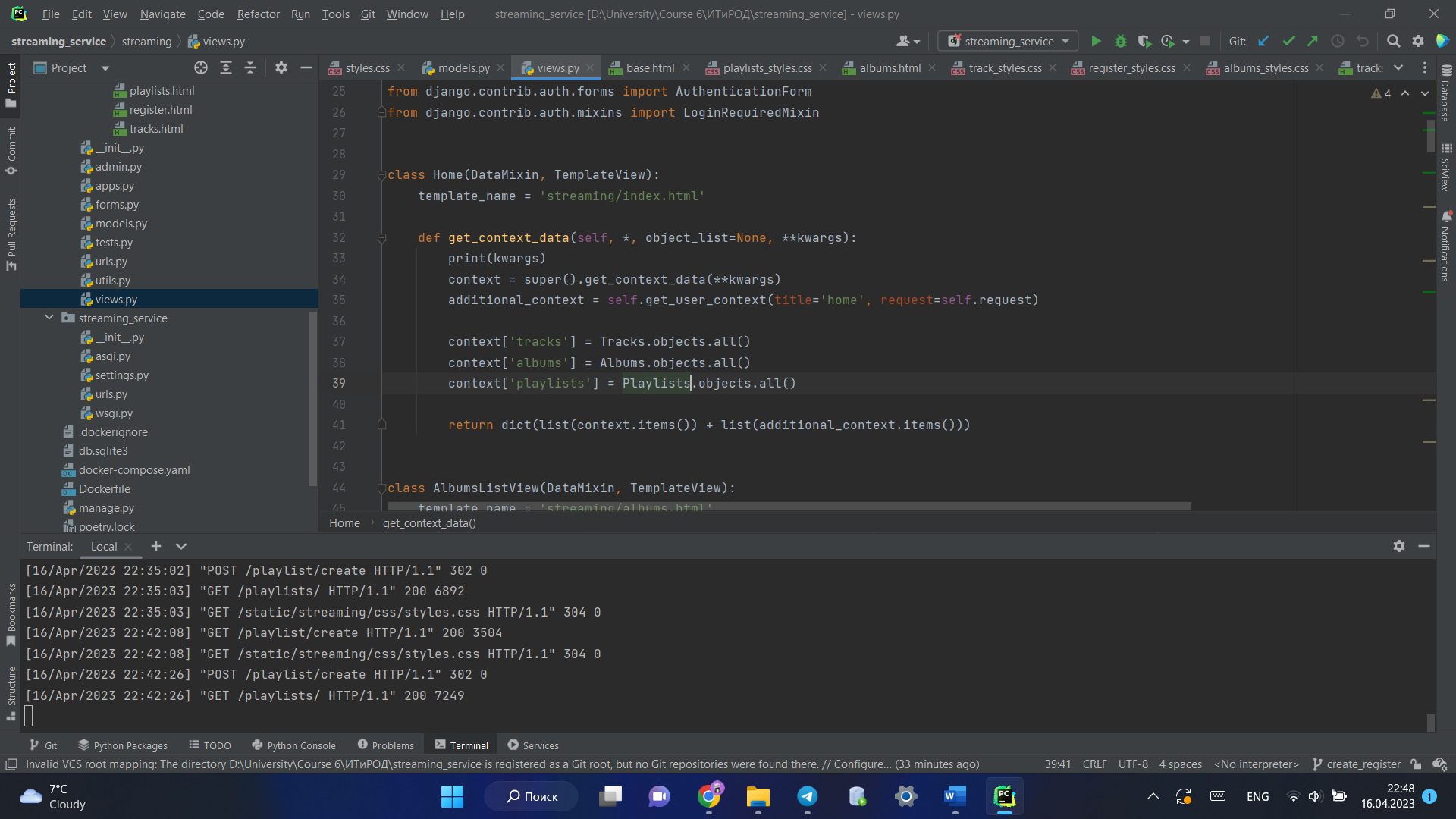


Рисунок 3.5 – View-class с выборкой данных

Также в программном продукте присутствуют 3 типа ролей:

* Обычный пользователь.

Данная роль дает доступ на просмотр всех данных: прослушивание треков, альбомов, плейлистов. (Рисунок 3.7)

* Артист.

Данная поль позволяет редактировать треки и альбомы со своим авторством, создавать новые, а также редактировать информацию о себе. (Рисунок 3.6)

* Администратор.

Имеет неограниченный доступ на изменение любых данных в системе через Django-admin панель. (Рисунок 3.8)

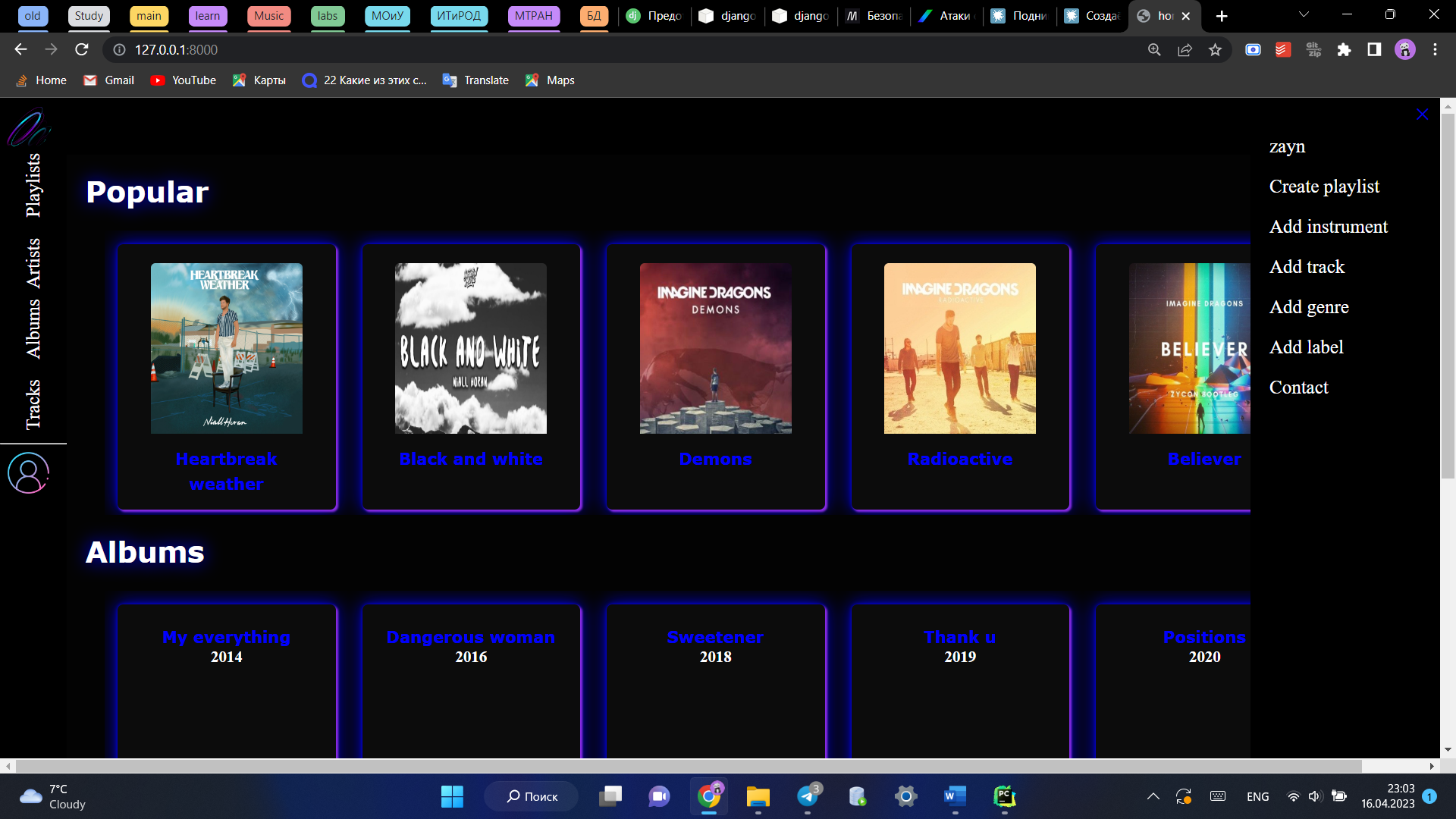


Рисунок 3.6 – Доступный функционал для роли “Артист”

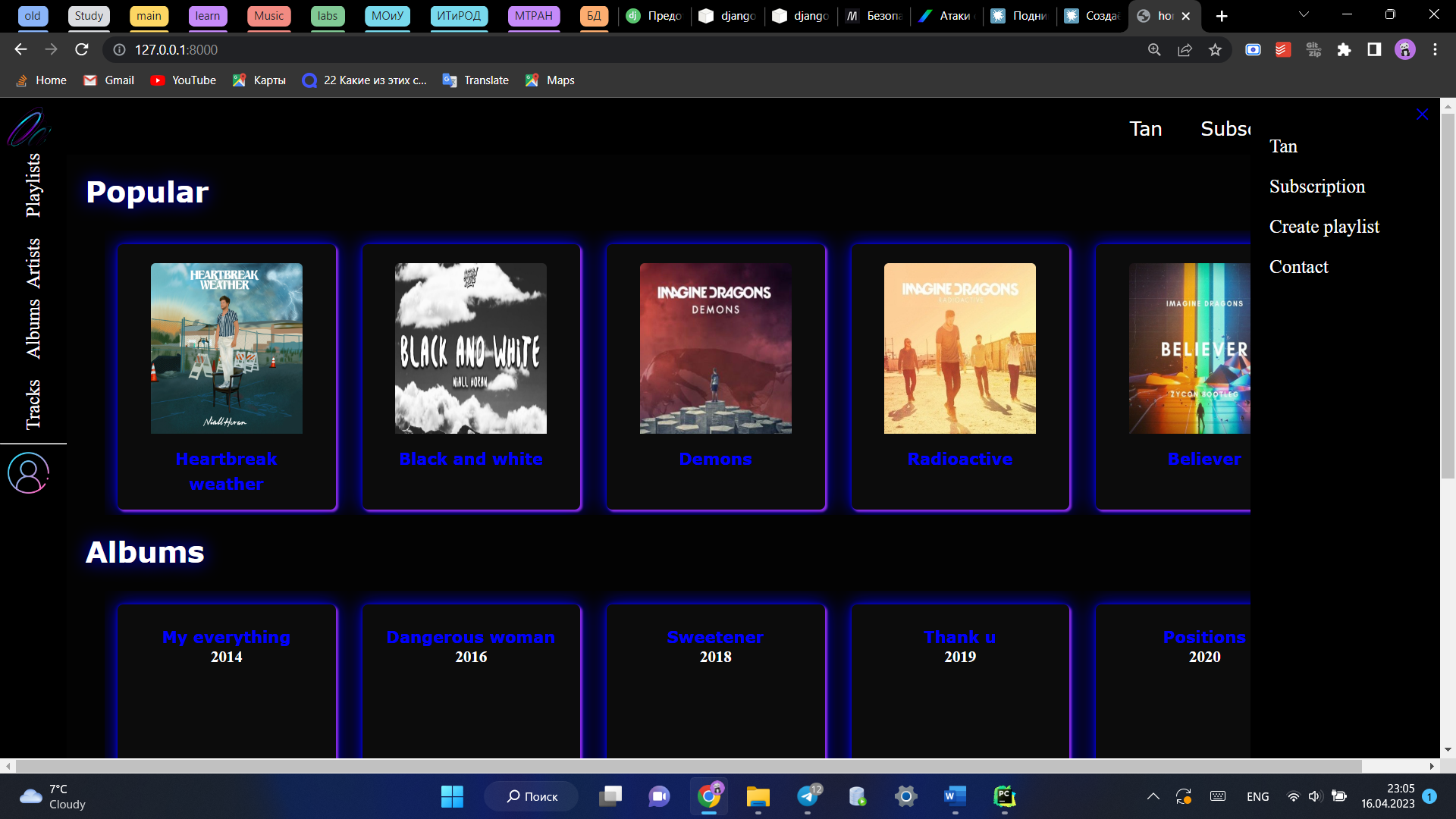


Рисунок 3.7 – Доступный функционал для роли “Пользователь”

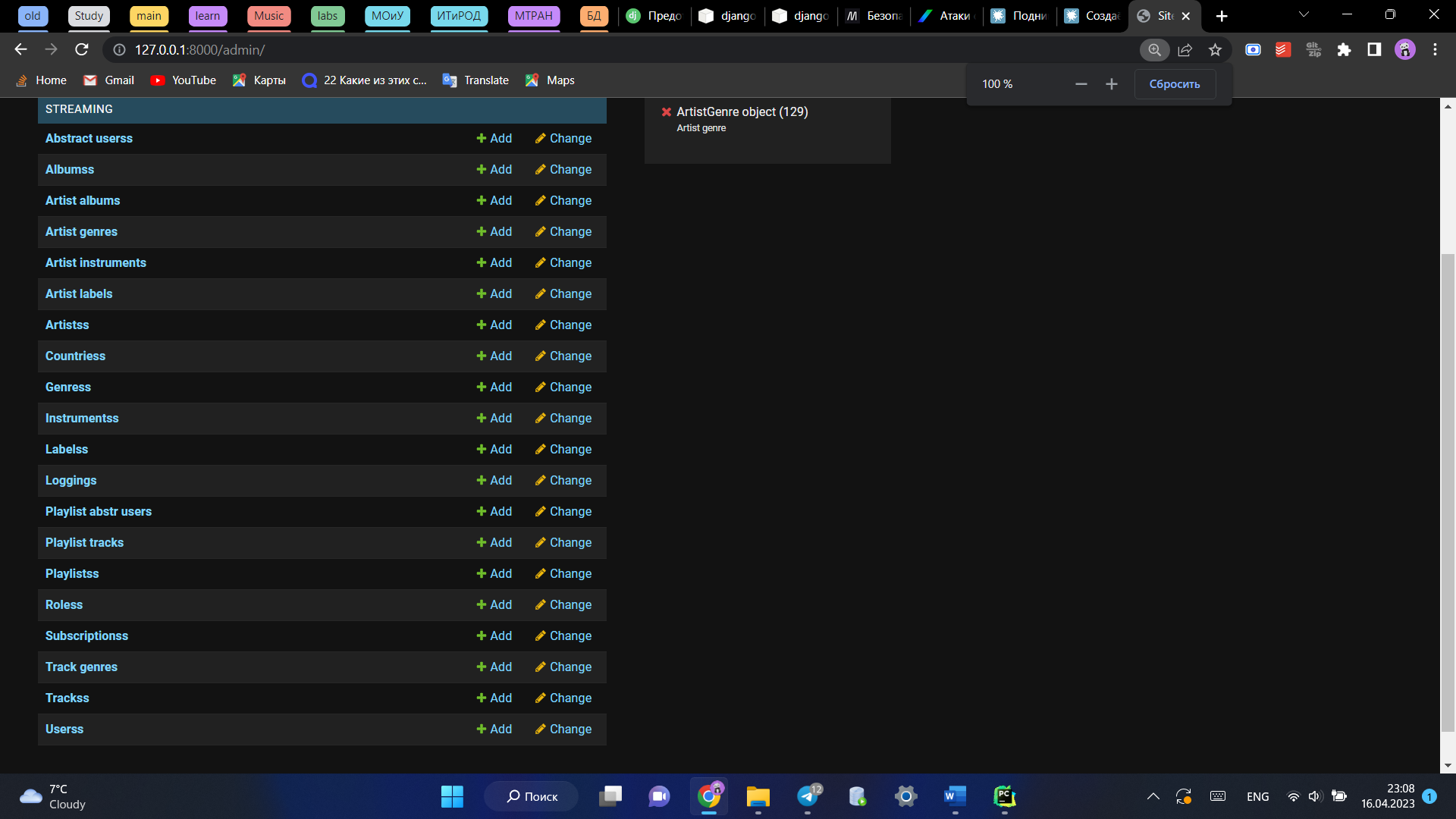


Рисунок 3.8 – Доступный функционал для роли “Администратор”

Защита от межсайтовой подделки запроса CSRF осуществляется путем добавления в шаблоны с формами, которые отправляются на сервер csrf\_tocken (Рисунок 3.9). Токен CSRF – это уникальное и непредсказуемое значение, которое сервер генерирует для защиты уязвимых перед CSRF ресурсов. После выполнения запроса приложение на стороне сервера сравнивает два маркера, найденные в сеансе пользователя и в самом запросе. Код формы в приложении А.

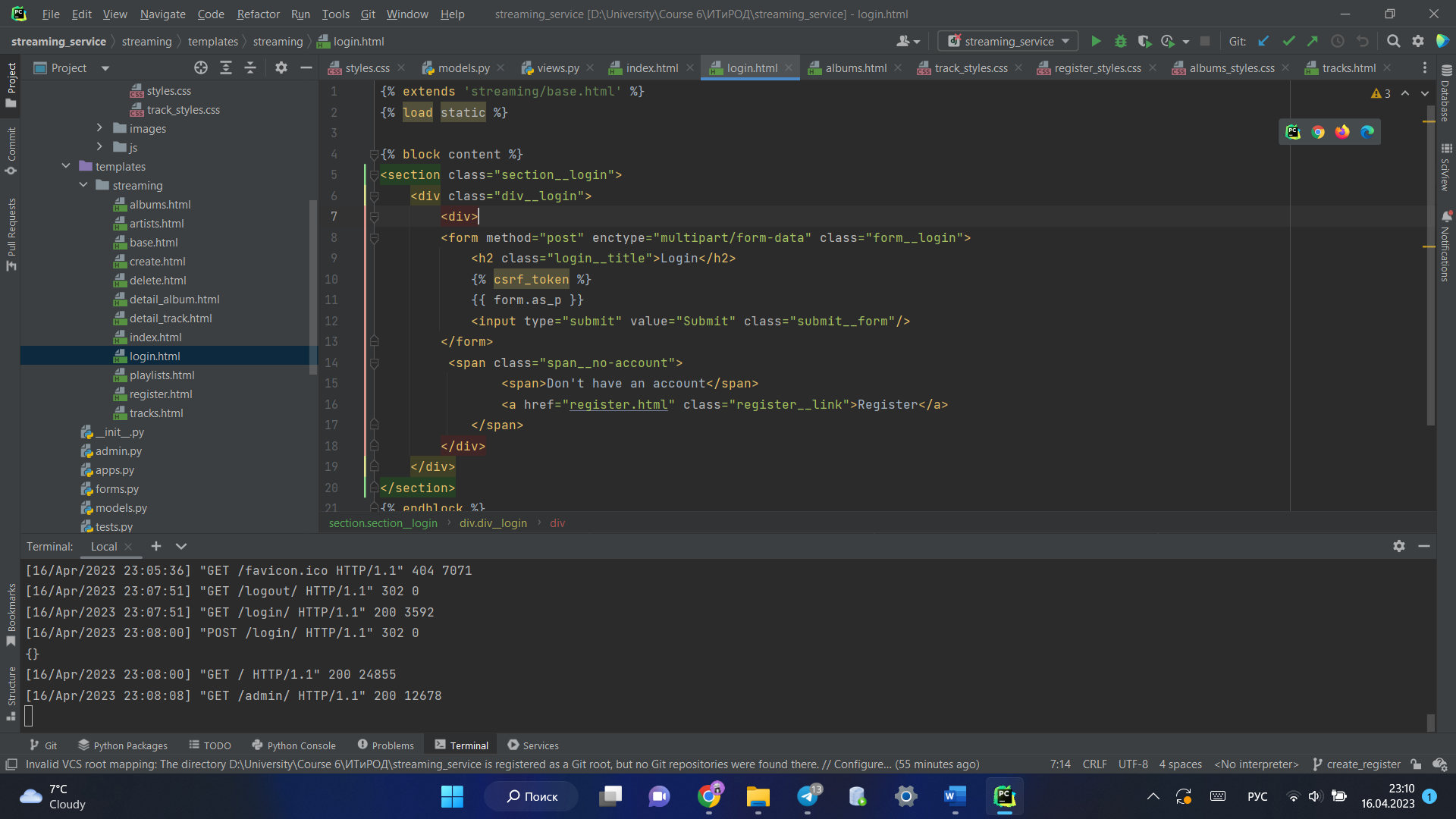


Рисунок 3.9 – Форма, при отправлении которой генерируется csrf\_tocken

На рисунках 3.10 – 3.13 процесс установки программы.

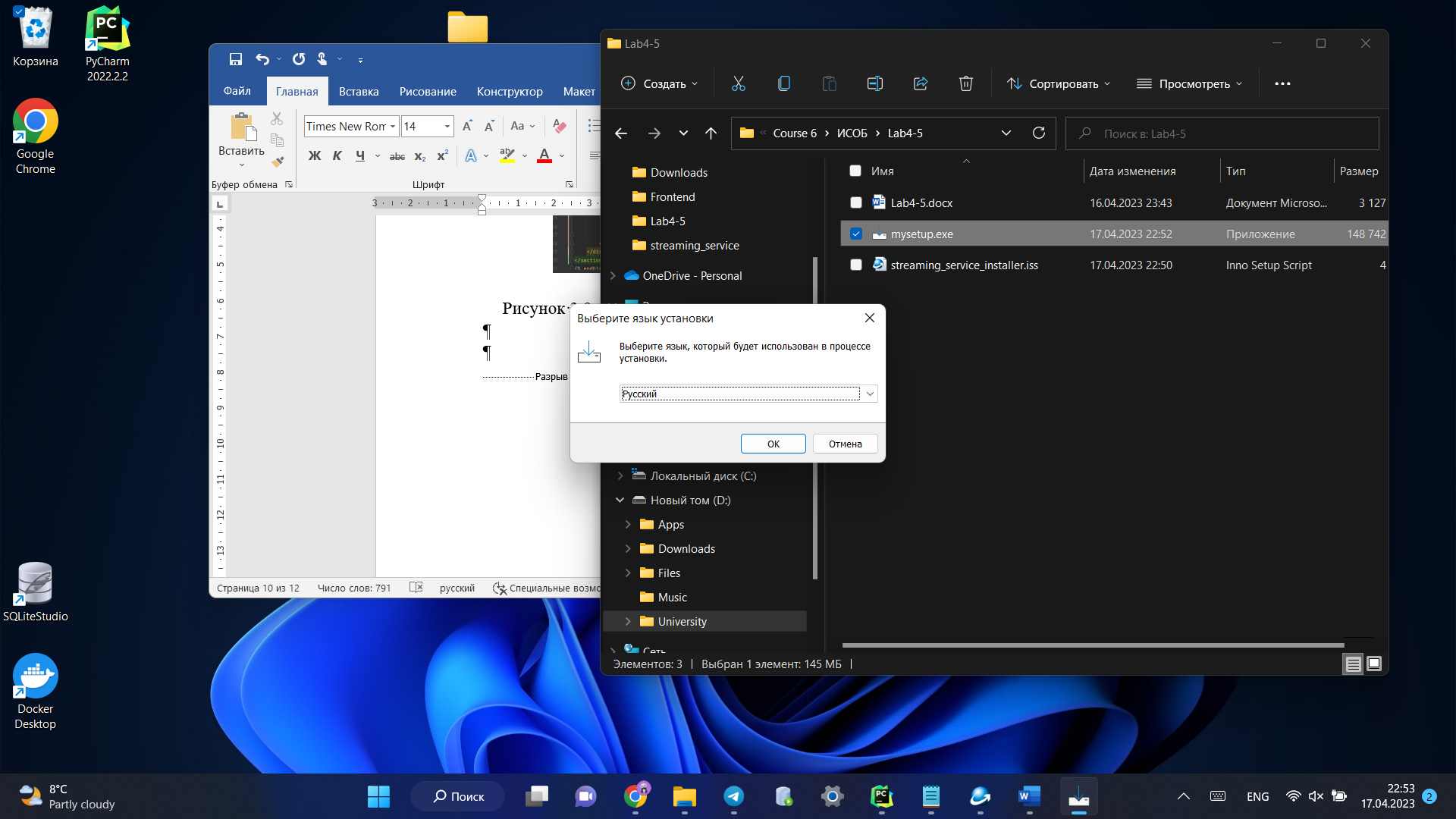


Рисунок 3.10 – Установка программы

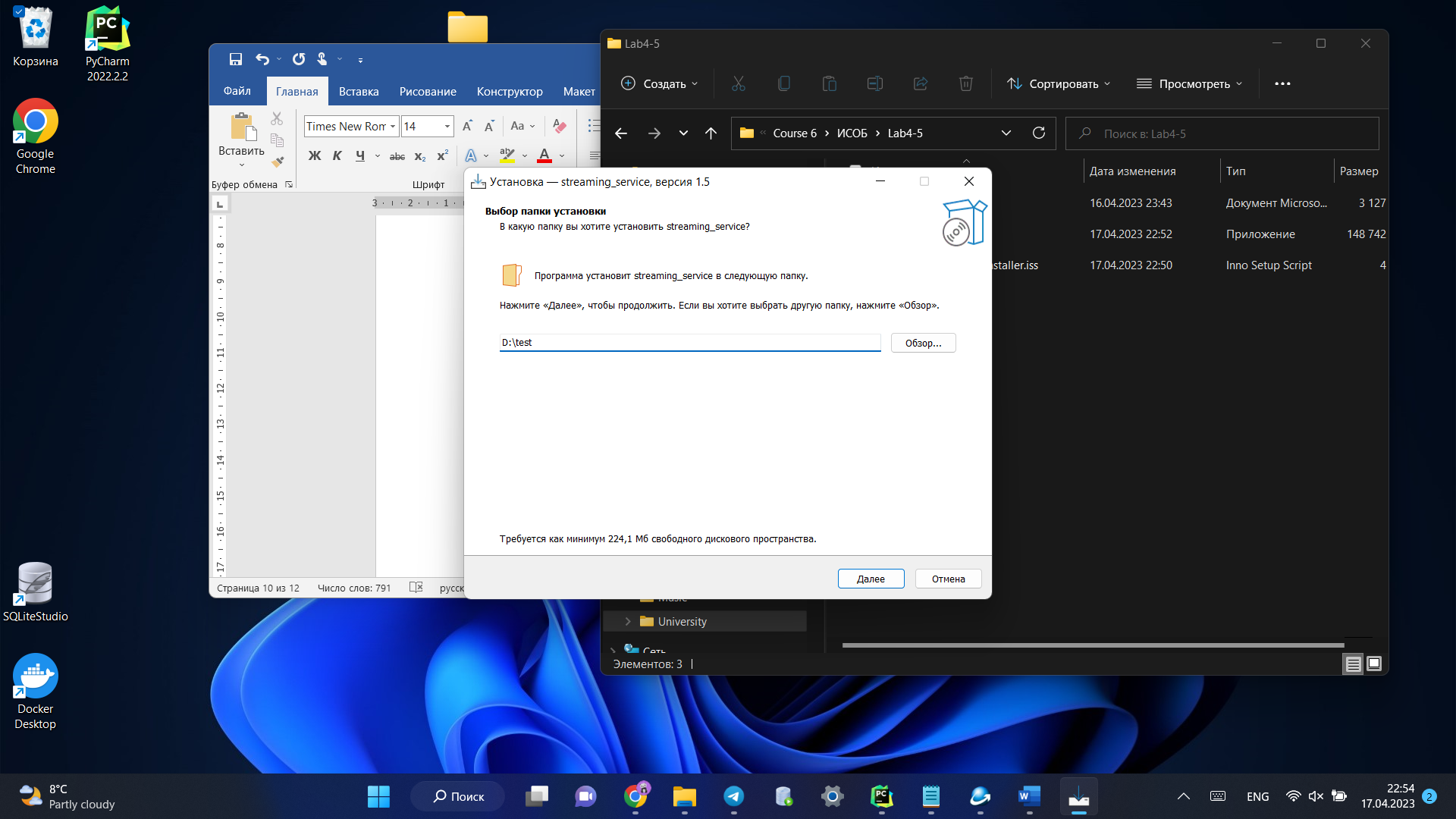


Рисунок 3.11 – Установка программы

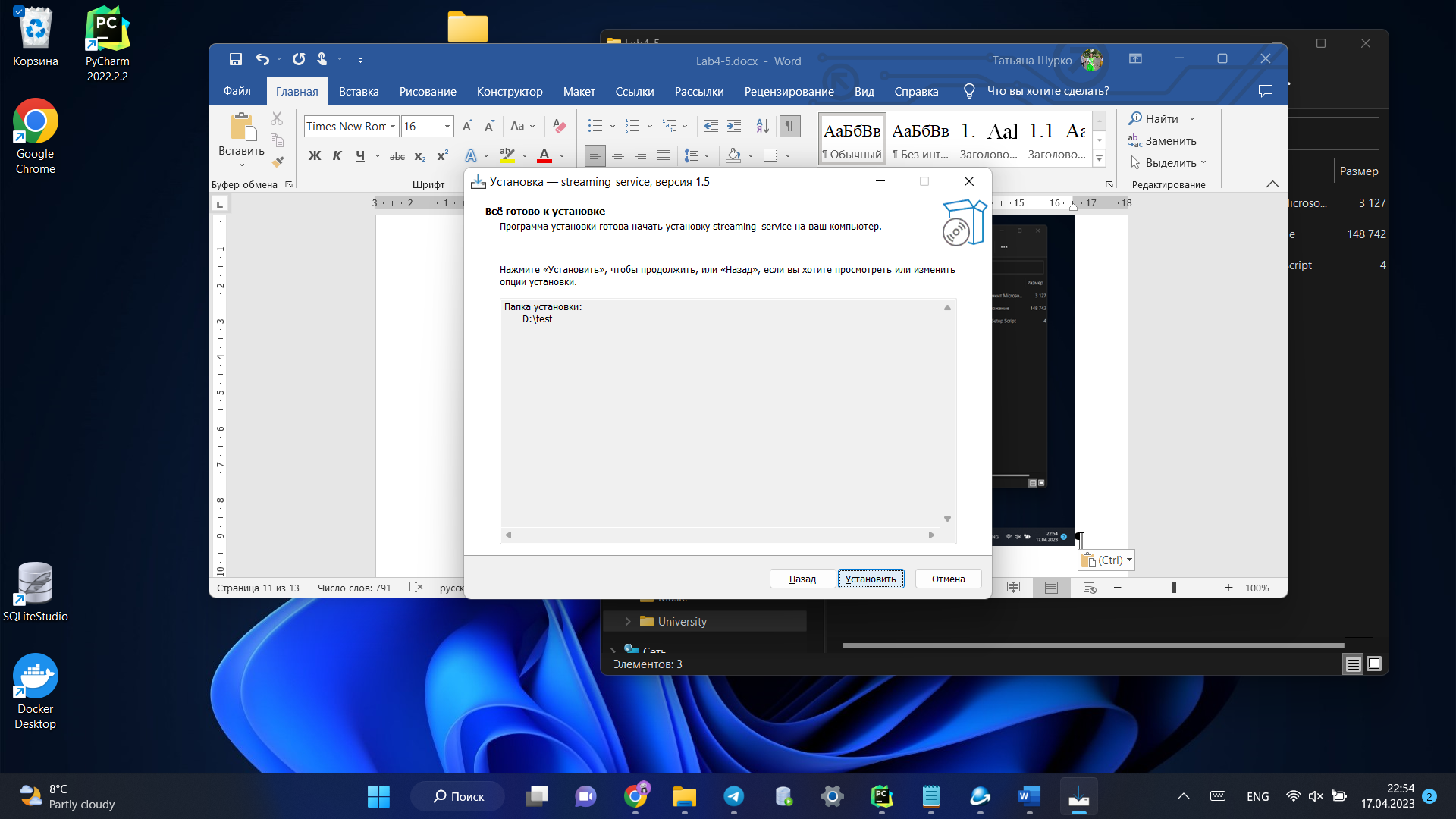


Рисунок 3.12 – Установка программы

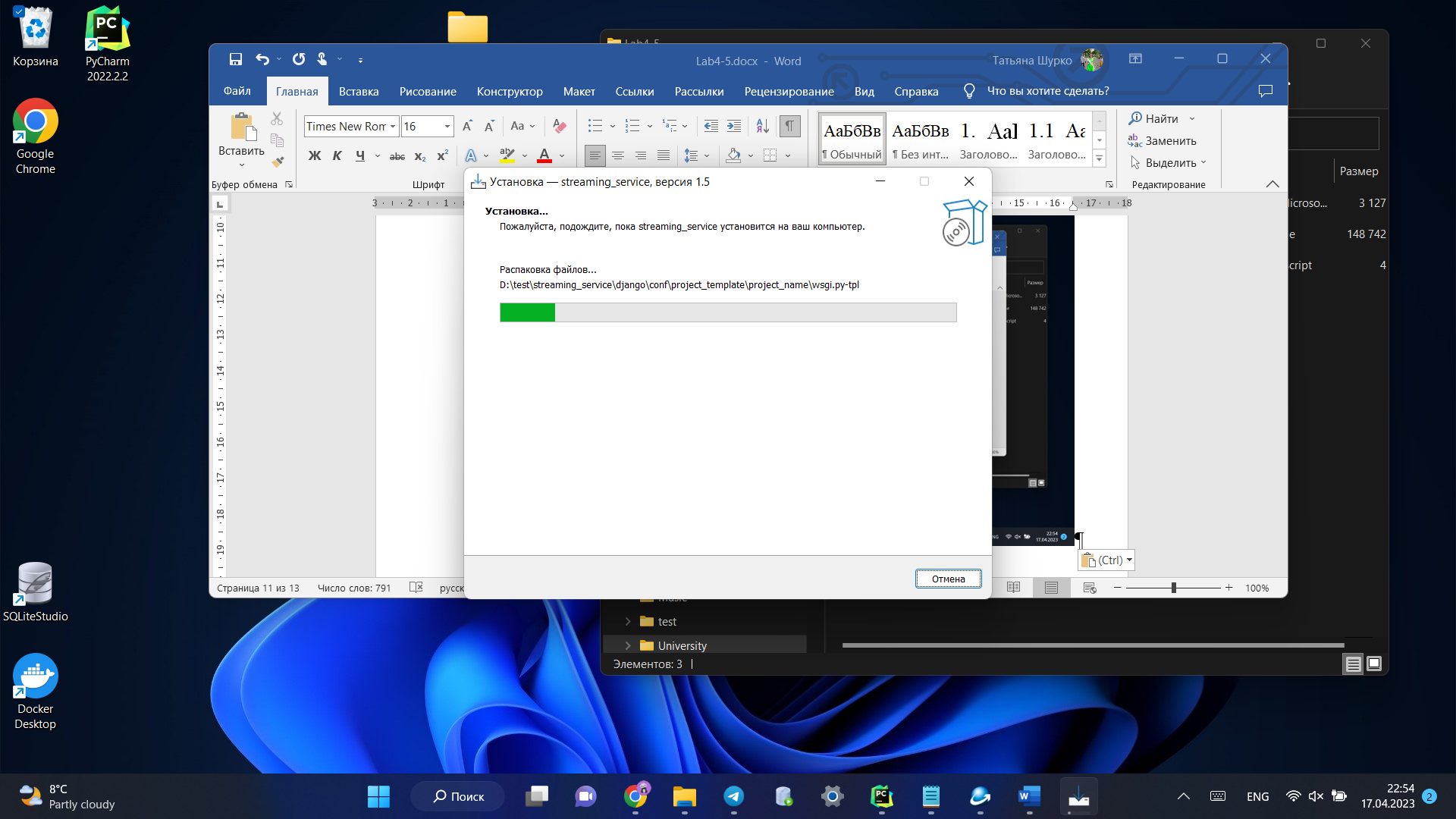


Рисунок 3.13 – Установка программы

# Вывод

В результате выполнения лабораторной работы №4, 5 ознакомились с концепцией ролевого управления доступом и способами защиты программного обеспечения от существующих угроз, научились разрабатывать приложения, которые используют ролевое управление доступом для разграничения полномочий пользователей, получили навыки защиты разработанной программы от несанкционированного копирования и других угроз, которым может подвергаться программное обеспечение.

Приложение А. Текст программы

{% extends 'streaming/base.html' %}

{% load static %}

{% block content %}

<section class="section\_\_login">

<div class="div\_\_login">

<div>

<form method="post" enctype="multipart/form-data" class="form\_\_login">

<h2 class="login\_\_title">Login</h2>

{% csrf\_token %}

{{ form.as\_p }}

<input type="submit" value="Submit" class="submit\_\_form"/>

</form>

<span class="span\_\_no-account">

<span>Don't have an account</span>

<a href="register.html" class="register\_\_link">Register</a>

</span>

</div>

</div>

</section>

{% endblock %}