Для локального запуска Django проекта с HTTPS на Windows есть несколько вариантов. Один из наиболее простых и популярных — использовать mkcert для создания локального SSL-сертификата и Django для его конфигурации.

**Шаги для настройки HTTPS для локального Django проекта на Windows**

**1. Установка mkcert**

* Скачайте и установите mkcert:
  + Перейдите на [страницу релизов mkcert](https://github.com/FiloSottile/mkcert/releases).
  + Скачайте подходящий архив для Windows и распакуйте его.
  + Переместите исполняемый файл mkcert.exe в любую папку, которая указана в переменной окружения PATH (например, в C:\Windows\System32).
* Установите mkcert:

sh

* mkcert -install

**2. Создание SSL-сертификатов**

* Перейдите в директорию вашего Django проекта:

sh

 cd path/to/your/django/project

 Создайте SSL-сертификаты:

sh

* mkcert localhost
* В результате будут созданы файлы localhost.pem и localhost-key.pem.

**3. Настройка Django для использования SSL-сертификатов**

* Откройте файл settings.py вашего Django проекта и добавьте следующие строки:

python

 import os

import ssl

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

# Путь к вашим сертификатам

CERTIFICATE\_FILE = os.path.join(BASE\_DIR, 'localhost.pem')

KEY\_FILE = os.path.join(BASE\_DIR, 'localhost-key.pem')

 Настройте Django для использования SSL:

Создайте или обновите файл runserver\_plus.py в папке вашего проекта (в той же папке, что и manage.py):

python

 from django.core.management.commands.runserver import Command as runserver

class Command(runserver):

def add\_arguments(self, parser):

super().add\_arguments(parser)

parser.add\_argument(

'--cert', dest='cert', type=str,

help='Path to SSL certificate file'

)

parser.add\_argument(

'--key', dest='key', type=str,

help='Path to SSL key file'

)

def handle(self, \*args, \*\*options):

if options.get('cert') and options.get('key'):

import ssl

self.ssl\_options = {

'certfile': options.get('cert'),

'keyfile': options.get('key'),

'ssl\_version': ssl.PROTOCOL\_TLS

}

super().handle(\*args, \*\*options)

 Запустите сервер Django с поддержкой HTTPS:

sh

* python manage.py runserver\_plus --cert localhost.pem --key localhost-key.pem

Теперь ваш локальный Django проект будет доступен через HTTPS.

**Дополнительные рекомендации**

* **Автоматизация:** Для удобства, вы можете добавить команду запуска сервера с HTTPS в Makefile или в скрипт.
* **Телеграм бот:** Убедитесь, что в настройках вашего бота прописаны корректные URL с использованием https://.

Этот подход позволяет просто и быстро настроить HTTPS для локального Django проекта, что удовлетворяет требованиям Телеграм бота.

Похоже, вы не установили django-extensions, которая добавляет команду runserver\_plus. Давайте установим этот пакет и правильно его настроим.

**Шаги для установки и настройки django-extensions**

1. **Установка django-extensions:**

Откройте командную строку и выполните следующую команду:

sh

 pip install django-extensions

 **Добавление django-extensions в INSTALLED\_APPS:**

Откройте файл settings.py и добавьте django\_extensions в список INSTALLED\_APPS:

python

 INSTALLED\_APPS = [

...

'django\_extensions',

]

 **Создание runserver\_plus команды:**

Убедитесь, что файл runserver\_plus.py находится в правильной директории:

* Создайте директорию management/commands внутри вашего приложения (например, main):

markdown

 your\_project/

├── main/

│ ├── management/

│ │ ├── \_\_init\_\_.py

│ │ └── commands/

│ │ ├── \_\_init\_\_.py

│ │ └── runserver\_plus.py

 Вставьте следующий код в файл runserver\_plus.py:

python

* 
* from django.core.management.commands.runserver import Command as runserver
* class Command(runserver):
* def add\_arguments(self, parser):
* super().add\_arguments(parser)
* parser.add\_argument(
* '--cert', dest='cert', type=str,
* help='Path to SSL certificate file'
* )
* parser.add\_argument(
* '--key', dest='key', type=str,
* help='Path to SSL key file'
* )
* def handle(self, \*args, \*\*options):
* if options.get('cert') and options.get('key'):
* import ssl
* self.ssl\_options = {
* 'certfile': options.get('cert'),
* 'keyfile': options.get('key'),
* 'ssl\_version': ssl.PROTOCOL\_TLS
* }
* super().handle(\*args, \*\*options)

 **Запуск сервера с HTTPS:**

Теперь попробуйте запустить сервер с HTTPS еще раз:

sh

1. python manage.py runserver\_plus --cert localhost.pem --key localhost-key.pem

**Проверка**

Если все настроено правильно, сервер должен запуститься с поддержкой HTTPS. Если возникнут дополнительные ошибки или проблемы, дайте знать, и мы продолжим их решать.

Эта ошибка означает, что вам нужно установить библиотеку Werkzeug, которая требуется для использования команды runserver\_plus.

**Установка Werkzeug**

1. **Установите Werkzeug:**

Откройте командную строку и выполните следующую команду:

sh

 pip install werkzeug

Установите библиотеку **pyOpenSSL:**

sh

pip install pyOpenSSL

После установки попробуйте снова запустить сервер с HTTPS:

sh

python manage.py runserver\_plus --cert localhost.pem --key localhost-key.pem

Ошибку ERR\_CERT\_COMMON\_NAME\_INVALID можно исправить, если правильно настроить и доверить созданный сертификат вашему браузеру. Вот обновленный процесс с подробностями:

**1. Создание сертификатов с помощью mkcert**

Если вы уже создали сертификаты с помощью mkcert, то пропустите этот шаг. Если нет, выполните следующую команду в командной строке:

sh

mkcert -install

mkcert 127.0.0.1 localhost

Эта команда создаст два файла: 127.0.0.1+1.pem и 127.0.0.1+1-key.pem.

**2. Настройка Django для использования новых сертификатов**

Откройте файл settings.py и убедитесь, что пути к сертификатам указаны правильно:

python

import os

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

CERTIFICATE\_FILE = os.path.join(BASE\_DIR, '127.0.0.1+1.pem')

KEY\_FILE = os.path.join(BASE\_DIR, '127.0.0.1+1-key.pem')

**3. Доверие сертификату в браузере**

Чтобы браузер доверял созданному сертификату, нужно его установить в доверенные сертификаты:

**Для Google Chrome или Microsoft Edge:**

1. Откройте chrome://settings/security (или edge://settings/security) в адресной строке браузера.
2. Прокрутите вниз и найдите раздел "Управление сертификатами" или "Certificates".
3. Нажмите на кнопку "Импортировать" (или "Import").
4. Следуйте инструкциям мастера импорта, выберите файл сертификата rootCA.pem, который создается mkcert (обычно находится в папке ~/.local/share/mkcert или C:\Users\<username>\AppData\Local\mkcert).
5. Убедитесь, что сертификат установлен в хранилище доверенных корневых сертификатов.

**Для Mozilla Firefox:**

1. Откройте about:preferences#privacy в адресной строке браузера.
2. Прокрутите вниз до раздела "Certificates" и нажмите на "View Certificates".
3. В открывшемся окне выберите вкладку "Authorities".
4. Нажмите "Import..." и выберите файл rootCA.pem.
5. Установите галочку "Trust this CA to identify websites".

**4. Перезапуск сервера**

Перезапустите ваш Django сервер с новыми сертификатами:

sh

python manage.py runserver\_plus --cert 127.0.0.1+1.pem --key 127.0.0.1+1-key.pem

**Проверка**

Теперь, когда вы откроете https://127.0.0.1:8000 в вашем браузере, он должен доверять сертификату и открывать сайт без ошибок.