**Тема занятия № 54: Модуль 31. Понятие "Токен". JSON Web Token Authentication**

1. Библиотека JWT – json web token.

Django поставляется с системой аутентификации, основанной на сеансах, и это работает из коробки. Это включает в себя все модели (models), представления (views) и шаблоны (templates), которые могут быть нужны вам для создания и дальнейшего логина пользователей. Но вот в чем загвоздка: стандартная система аутентификации Django работает только с традиционным 'запрос-ответ' циклом HTML.

Что мы имеем ввиду под «традиционным 'запрос-ответ' циклом HTML»? Исторически, когда пользователь хотел выполнить какое-то действие (например, создать новый аккаунт), он заполнял определенную форму в браузере. Далее, когда он кликал на кнопку «Отправить», браузер формировал запрос - который включал в себя данные, введенные пользователем - и отправлял на сервер, сервер обрабатывал запрос, и отвечал либо HTML страницей, либо редиректом на новую страницу. Это то, что мы имеем ввиду, когда говорим о «полном обновлении страницы».

Почему важно знать, что встроенная система аутентификации Django работает только с традиционным 'запрос-ответ' циклом HTML? Потому что клиент, для которого мы создадим данный API, не придерживается этого цикла. Вместо этого, клиент будет ожидать, что сервер вернет JSON, вместе обычного HTML. Возвращая JSON, мы можем позволить решать клиенту, а не серверу, что делать дальше. В цикле 'запрос-ответ' JSON, сервер получает данные, обрабатывает их и возвращает ответ (пока что как и в цикле 'запрос-ответ' HTML), но ответ не управляет поведением браузера. Ответ просто сообщает браузеру результат запроса.

К счастью, команда разработки Django поняла, что тренды веб разработки движутся именно в этом направлении. Они также знали, что некоторые проекты могут не захотеть использовать встроенные модели, представления и шаблоны. Вместо этого, они могут использовать собственные. Чтобы убедиться, что все усилия, затраченные на создание встроенной системы аутентификации Django, не потрачены зря, они решили сделать возможным использование наиболее важных частей, сохраняя при этом возможность настройки конечного результата.

Аутентификация, основанная на токенах наиболее распространенной альтернативой аутентификации на основе сессий/сеансов, является т.н. аутентификация на основе токенов. Мы будем использовать особую форму такой аутентификации для защиты нашего приложения. При аутентификации на основе токенов сервер предоставляет клиенту токен после успешного запроса на вход. Этот токен уникален для пользователя и хранится в базе данных вместе идентификатором пользователя (если точнее, возможны раные вариации генерации токена, основная же идея в том, чтобы он аутентифицировал пользователя, позволяя знать кто это и давая доступ к апи, и имел время жизни, по истечении которого "протухал"). Ожидается, что клиент отправит токен вместе с будущими запросами, чтобы сервер мог идентифицировать пользователя. Сервер делает это путем поиска в таблице базы данных, содержащей все созданные токены. Если соответствующий токен найден, то сервер продолжает проверять, действителен ли токен. Если не найден, мы говорим, что пользователь не аутентифицирован. Поскольку токены хранятся в базе данных, а не в файлах куки, аутентификация на основе токенов соответствует нашим потребностям.

**Верификация токенов**

Мы всегда имеем возможность сохранить не только идентификатор пользователя (ID) с его токеном. Мы также можем хранить такие вещи, как дата истечения срока действия токена. В данном примере нам необходимо убедиться, что срок действия токена не прошел. Если прошел - считать, что токен недействителен. В таком случае, мы удаляем его из базы данных и просим пользователя снова войти в систему.

**JSON Web Tokens**

JSON Web Token (сокр. JWT) - это открытый стандарт (RFC 7519), который определяет компактный и автономный способ безопасной передачи информации между двумя сторонами. Можно думать о JWT как о токенах аутентификации на стероидах.

Помните, что мы сказали, что будем использовать особую форму аутентификации на основе токенов? JWT это как раз то, что имелось ввиду.

**Почему JSON Web Tokes лучше обычных токенов?**

При переходе с обычных токенов на JWT мы получаем несколько преимуществ:

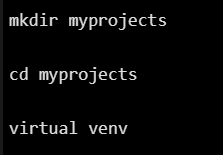
JWT - открытый стандарт. Это означает, что что все реализации JWT должны быть довольно похожими, что является преимуществом при работе с разными языками и технологиями. Обычные токены имеют более свободную форму, что позволяет разработчику решать, как лучше всего реализовывать токены.

JWT содержат информацию о пользователе, что удобно для клиентской стороны.

Библиотеки здесь берут на себя основную тяжелую работу. Развертывание собственной системы аутентификации опасно, поэтому мы оставляем важные вещи проверенным «в боях» библиотекам, которым можем доверять.

Давайте начнем.

Создайте каталог, в котором вы будете содержать свой проект, а также виртуальную среду для установки зависимостей проекта.

****

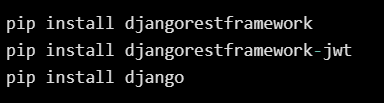
Активируйте виртуальную среду:



Создайте проект Django.



Установите DRF и django-rest-framework-jwt, используя pip.

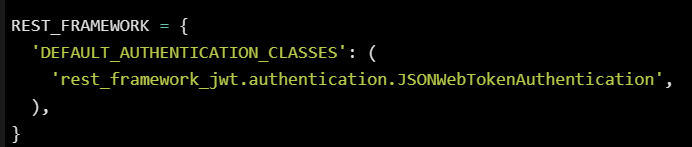
****

Давайте продолжим и добавим DRF в список установленных приложений в файле settings.py.

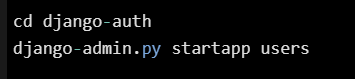
**Настройка параметров JWT**

Чтобы использовать JWT, нам нужно настроить разрешения django-rest-framework для принятия JSON Web Tokens.

В файле settings.py добавьте следующие конфигурации:

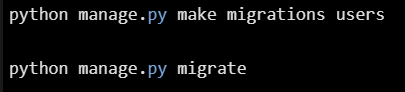
****

Создайте новое приложение users, которое будет обрабатывать аутентификацию и управление пользователями.

****

Добавьте приложение users в список установленных приложений в файле settings.py.

Создайте первоначальную миграцию для модели наших пользователей и выполните синхронизацию базы данных.

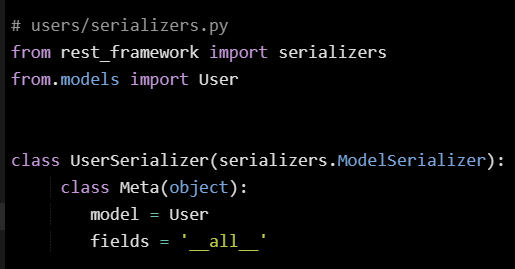
****

**Создание суперпользователя**

Создайте суперпользователя, выполнив следующую команду:



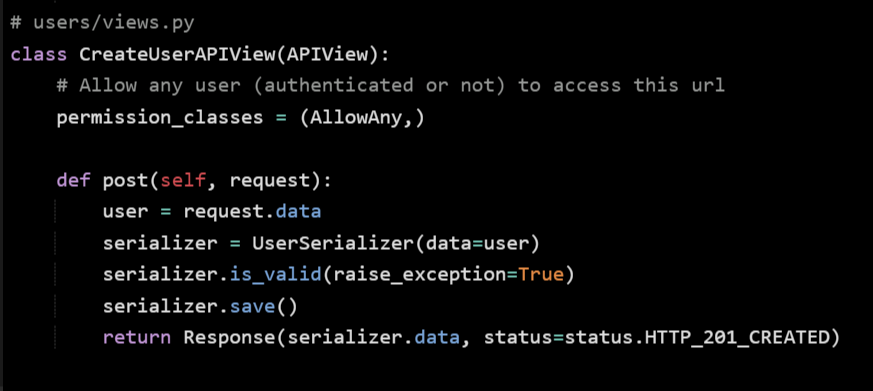
Создайте users/serializers.py и добавьте следующий код.

****

**CreateUserAPIView**

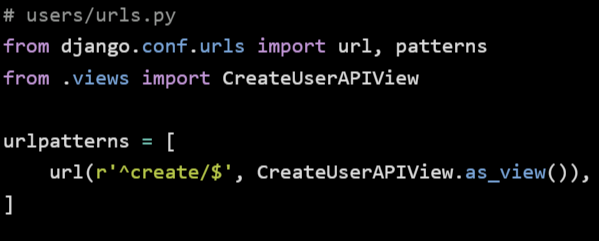
Затем мы хотим создать представление, чтобы у клиента был URL-адрес для создания новых пользователей.

В user.views.py добавьте следующее:

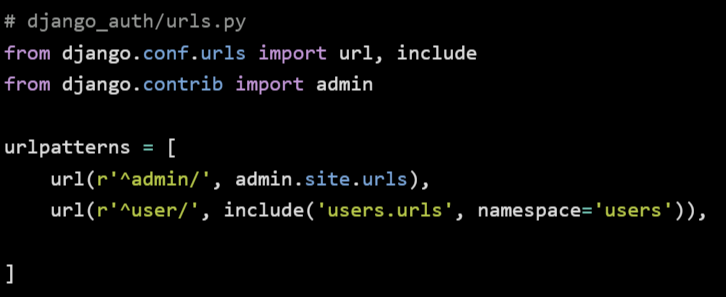
****

**Настройка URL-адресов**

Создайте файл users/urls.py и добавьте URL-адрес в соответствие с созданным нами представлением. Также добавьте следующий код.

****

Нам также нужно импортировать URL-адреса из приложения-пользователя в основной файл django\_auth/urls.py. Так что делаем это и двигаемся дальше. Мы используем здесь функцию include, поэтому не забудьте ее импортировать.

****

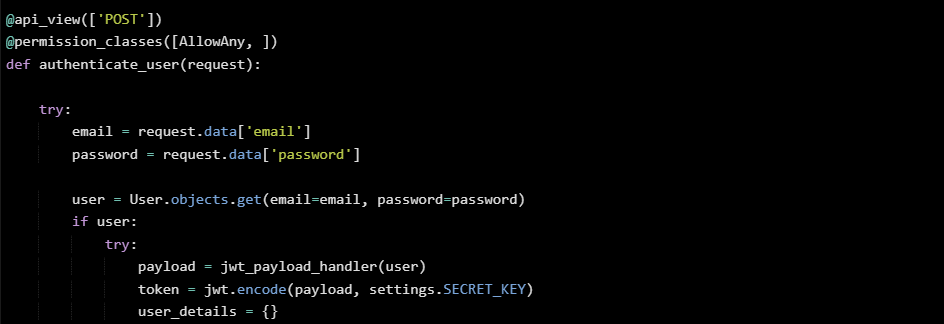
**Аутентификация пользователей**

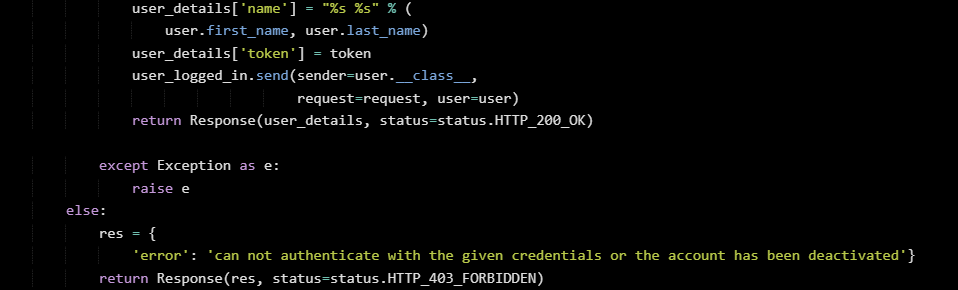
Мы будем использовать модуль Django-REST Framework JWT Python, который мы установили в начале этого руководства. Он добавляет поддержку JWT-аутентификации для приложений Django Rest Framework.

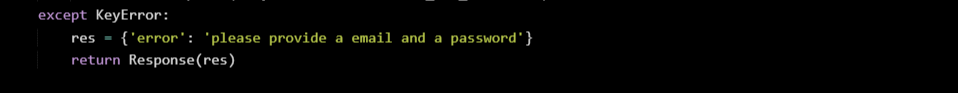
Но сначала давайте определим некоторые параметры конфигурации для наших токенов и как они сгенерированы в файле settings.py.

****

В users/views.py добавьте следующий код.



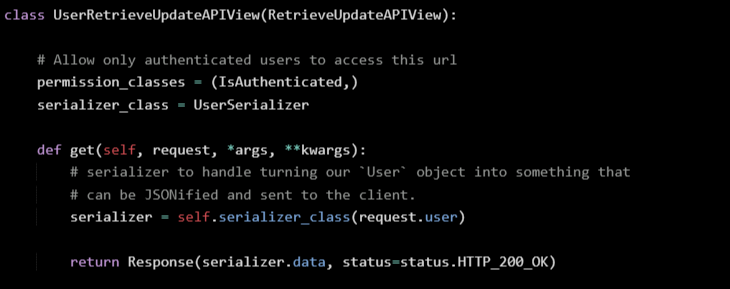


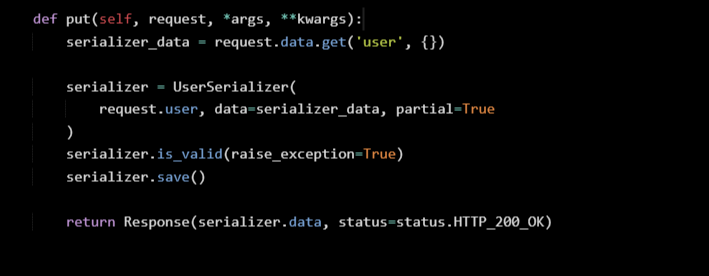


**Получение и обновление пользователей**

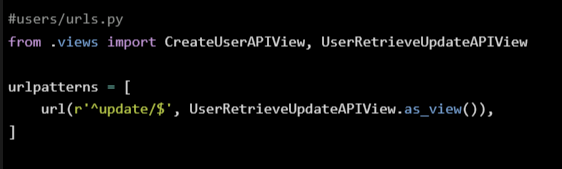
Пока пользователи могут зарегистрироваться и пройти проверку подлинности. Тем не менее, так же нужно предоставить способ получения и обновления этой информации. Давайте реализуем это.

В файле user.views.py добавьте следующий код.





Обновите URL-адреса, чтобы определить ендпоинт следующим образом.

****

Чтобы запрос был успешным, заголовки должны содержать JWT токен.