**Глава 4(Создание представления входа в систему)**

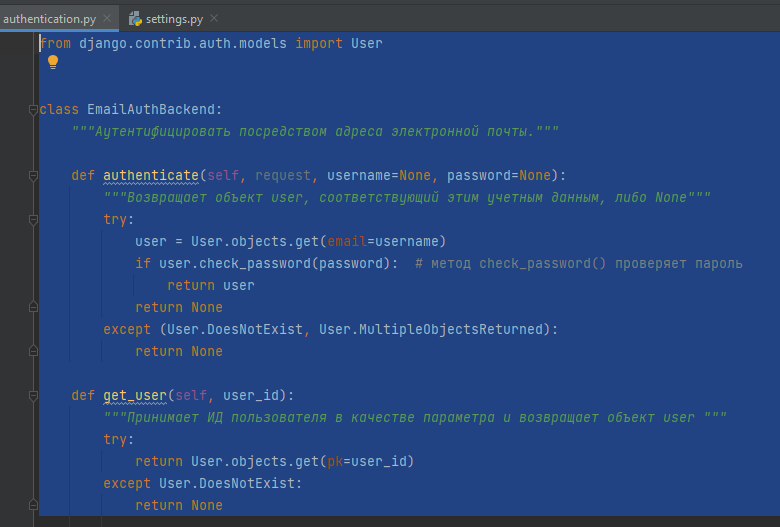
**authenticate()** = **Аутентифицирует** пользователя по базе данных. Указанный метод принимает объект request, параметры username и password и возвращает объект User, если пользователь был успешно аутентифицирован, либо **None** в противном случае.

**is\_active =** Проверяет активен ли пользователь

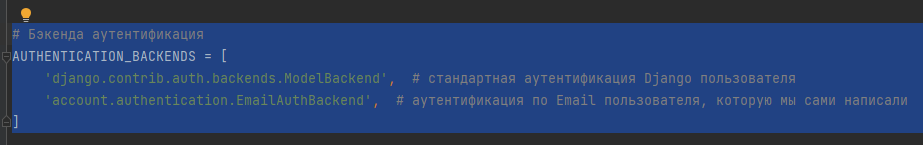
**login()** = Задает пользователя в текущем сеансе для входа пользователя в сайт. ВХОД выполняется вызовом функции **login**. Запрос должен быть представлен экземпляром класса HttpRequest, а пользователь, от имени которого выполняется вход, — записью модели User.

**Глава 4(Разработка конкретно-прикладного бэкенда аутентификации)**

Мы создали **дополнительный** бэкенд **аутентификацию**, для этого мы создали файл **authentication.py** и в нем прописали класс аутентификации



После в настройках проекта добавили собственную аутентификацию после встроенного, и теперь пользователь может аутентифицироваться как по **имени**, так и по **email**



**Глава 4(**Регистрация пользователя**)**

В форме если нужно проверить валидацию отдельного поля можно использовать метод с началом **clean\_<название поля>**. Такие методы исполняются, когда форма проходит валидацию при вызове метода is\_valid(). А если хотите проверить/валидировать всю форму можно использовать метод **clean()**

Пример:

