Практическое занятие №3

## Задача 1: Расширение пользовательской модели.

В Django пользователь по умолчанию имеет ограниченной набор атрибутов. Почти в любом проекте возникает необходимость добавить новых атрибутов пользователю. Существует 4 способа сделать это:

#### Простое расширение модели (proxy)

Эта стратегия без создания новых таблиц в базе данных. Используется, чтобы изменить поведение существующей модели (например, упорядочение по умолчанию, добавление новых методов и т.д.), не затрагивая существующую схему базы данных.

Вы можете использовать эту стратегию, когда вам не нужно хранить дополнительную информацию в базе данных, а просто необходимо добавить дополнительные методы или изменит диспетчер запросов модели.

#### Использование связи один-к-одному с пользовательской моделью (user profiles)

Это стратегия с использованием дополнительной обычный модели Django со своей таблицей в базе данных, которая связана пользователем стандартной модели через связь OneToOneField.

Вы можете использовать эту стратегию, чтобы хранить дополнительную информацию, которая не связана с процессом аутентификации (например, дата рождения). Обычно это называется пользовательский профиль. →

#### Расширение AbstractBaseUser

Это стратегия использования совершенно новой модели пользователя, которая отнаследована от AbstractBaseUser. Требует особой осторожности и изменения настроек в settings.py. В идеале должно быть сделано в начале проекта, так как будет существенно влиять на схему базы данных.

Вы можете использовать эту стратегию, когда ваш сайт имеет специфические требования в отношении процесса аутентификации. Например, в некоторых случаях имеет смысл использовать адрес электронной почты в качестве идентификации маркера вместо имени пользователя.

#### Расширение AbstractUser

Это стратегия использования новой модели пользователя, которая отнаследована от AbstractUser. Требует особой осторожности и изменения настроек в settings.py. В идеале должно быть сделано в начале проекта, так как будет существенно влиять на схему базы данных.

Вы можете использовать эту стратегию, когда сам процесс аутентификации Django вас полностью удовлетворяет и вы не хотите его менять. Тем не менее, вы хотите добавить некоторую дополнительную информацию непосредственно в модели пользователя, без необходимости создавать дополнительный класс (как в варианте 2)

Подробнее о каждом из описанных спеособов можно поучитать тут [<https://habr.com/ru/post/313764/#AbstractUser>]

К лучшим практимкам следует отнести расширение средствами AbstractUser.

### AbstractUser

Следующий пример показывает, как добавить новые атрибуты для стандартного пользователя Django.

Необходимо создать таблицу модели для хранения новых полей пользователя.

В models.py:

|  |
| --- |
| from django.db import models from django.contrib.auth.models import AbstractUser  class User(AbstractUser):  name = models.CharField(max\_length=100, blank=True, null=True) |

Вторым этапом необходимо объявить созданную модель в settings.py:

|  |
| --- |
| AUTH\_USER\_MODEL = 'your\_app.User' |

Для использования своей модели юзера в других таблицах в файле models.py, необходимо сослаться на эту модель с помощью `settings.AUTH\_USER\_MODEL` или` get\_user\_model () `модуля авторизации Django.

|  |
| --- |
| from django.conf import settings  from django.contrib.auth import get\_user\_model  class Book(models.Model):  author = models.ForeignKey(settings.AUTH\_USER\_MODEL) (OR)  from django.contrib.auth import get\_user\_model  User = get\_user\_model()  class Book(models.Model):  author = models.ForeignKey(User) |

Задача: расширить пользователя владельца автомобиля, так, чтобы о нем хранилась следующая информация: номер паспорта, домашний адрес, национальность. Отобразить новые поля пользователя в Django Admin. Отредактировать код из предыдущих работа, так, чтобы выводилась информация о пользователях.