1. **AbstractUser** – абстрактный класс модели пользователя.
   1. Наследуем конкретный класс User, который будем использовать во всем приложении.
      1. Роль будет определяться выбором в выпадающем меню
         1. Студент
         2. Преподаватель
         3. Гость (не факт, что будем вводить)
      2. Возможен вариант распределения ролей через отдельную таблицу, путем прямого назначения прав.
         1. Т.е. будет группа «Студент» -> Список прав
         2. Группа «Преподаватель» -> Список прав
         3. И т.д.
   2. Возможен вариант наследования отдельного класса Student и Teacher
   3. Или наследовать Student и Teacher через proxy (это будет 1 в 1 как класс User, только с доп. Фильтрацией).
2. **StudyGroup** – модель учебной группы:
   1. Название
   2. Дата создания
   3. Год – думаю можно оставить
   4. Студенты. \*В каждой группе может быть много студентов, но студент может быть только в одной группе. 1 -> N (**внешний ключ в модели Студента**)
   5. Преподаватели. \*У одной группы может быть много преподавателей, и преподаватель может быть в нескольких группах. (многое-ко-многим, ключ в модели **Преподавателя**)
   6. Доступные предметы (сделаем многое-ко-многим)
3. **StudySubjet** – класс модели учебного предмета:
   1. title – название (Теория вероятности, линейная алгебра и т.д.)
4. **TheoryCategory** – класс раздела теор. Материала:
   1. **Parent** – внешний ключ «сам-в-себя», может быть пустым.:  
      parent = models.ForeignKey('self',blank=**True**, null=**True** ,related\_name='children')
   2. **Subject –** внешний ключ на **StudySubject**
5. **TheoryPost** – класс самого теор. Материала:
   1. Content – аттрибут текста материала.
   2. **Subject –** внешний ключ на **StudySubject**
6. **PracticeCategory** – класс раздела практического. Материала:
   1. **Parent** – внешний ключ «сам-в-себя», может быть пустым.:  
      parent = models.ForeignKey('self',blank=**True**, null=**True** ,related\_name='children')
   2. **Subject –** внешний ключ на **StudySubject**
7. **PracticeTask** – класс самой задачи
   1. Content – аттрибут текста задачи.
   2. Complexity – аттрибут сложности
   3. Type – аттрибут типа
8. **PracticeType** – класс модели «Типа задачи»
9. **PractiveTaskVariation** – класс модели «Случайных параметров задачи»
   1. Task – внешний ключ на PractiveTask
   2. Аттрибуты с данными по случайным параметрам.
10. **UserProgress** – класс модели прогресса пользователя
    1. User
    2. Score
    3. answers

\*Категории повторяют друг друга, поэтому можно сделать абстрактный класс Category и от него наследовать все остальное.

\*Логика структуры задач:

* Есть базовый раздел, например, «параграф $1».
* Внутри существуют 3 типа задач:
  1. Тестовые задачи
  2. Задачи для самообучения
  3. Контрольные задачи
* Каждый раздел необходимо закрыть.
* Оценка выставляется за пройденный раздел.

Т.е. можно реализовать «Главную категорию», внутри добавить подкатегории «Тестовые задачи», «Контрольные задачи» и внутри уже будет содержаться список задач.

Как только решаются все задачи, раздел автоматически подсчитывает оценку.

Задачи могут допускать дополнительное кол-во попыток.

Все ответы пользователя необходимо записывать в БД.