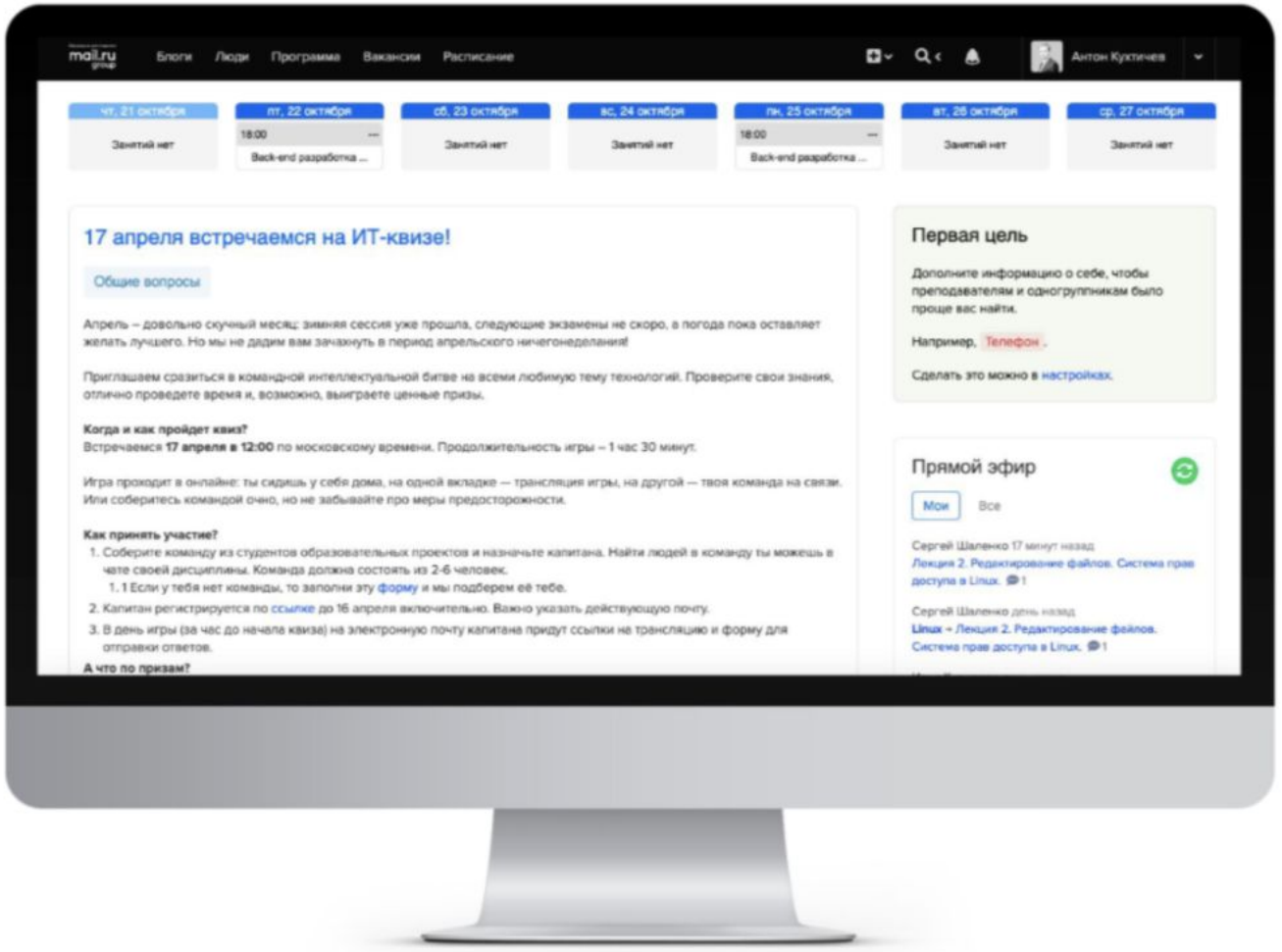
Бэкенд-разработка на Python. Лекция №6. ORM

Алена Елизарова

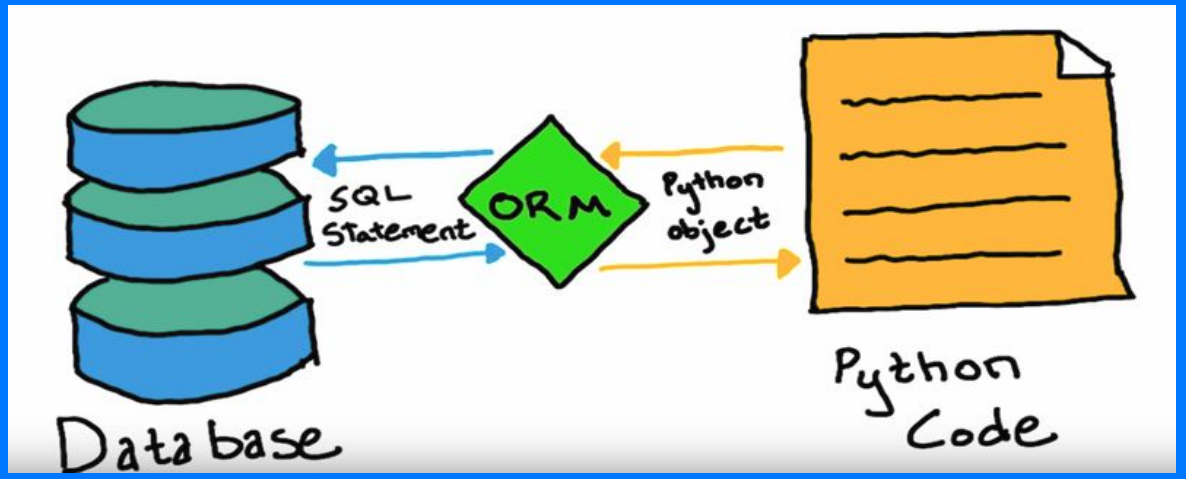
образование

Напоминание отметиться на портале



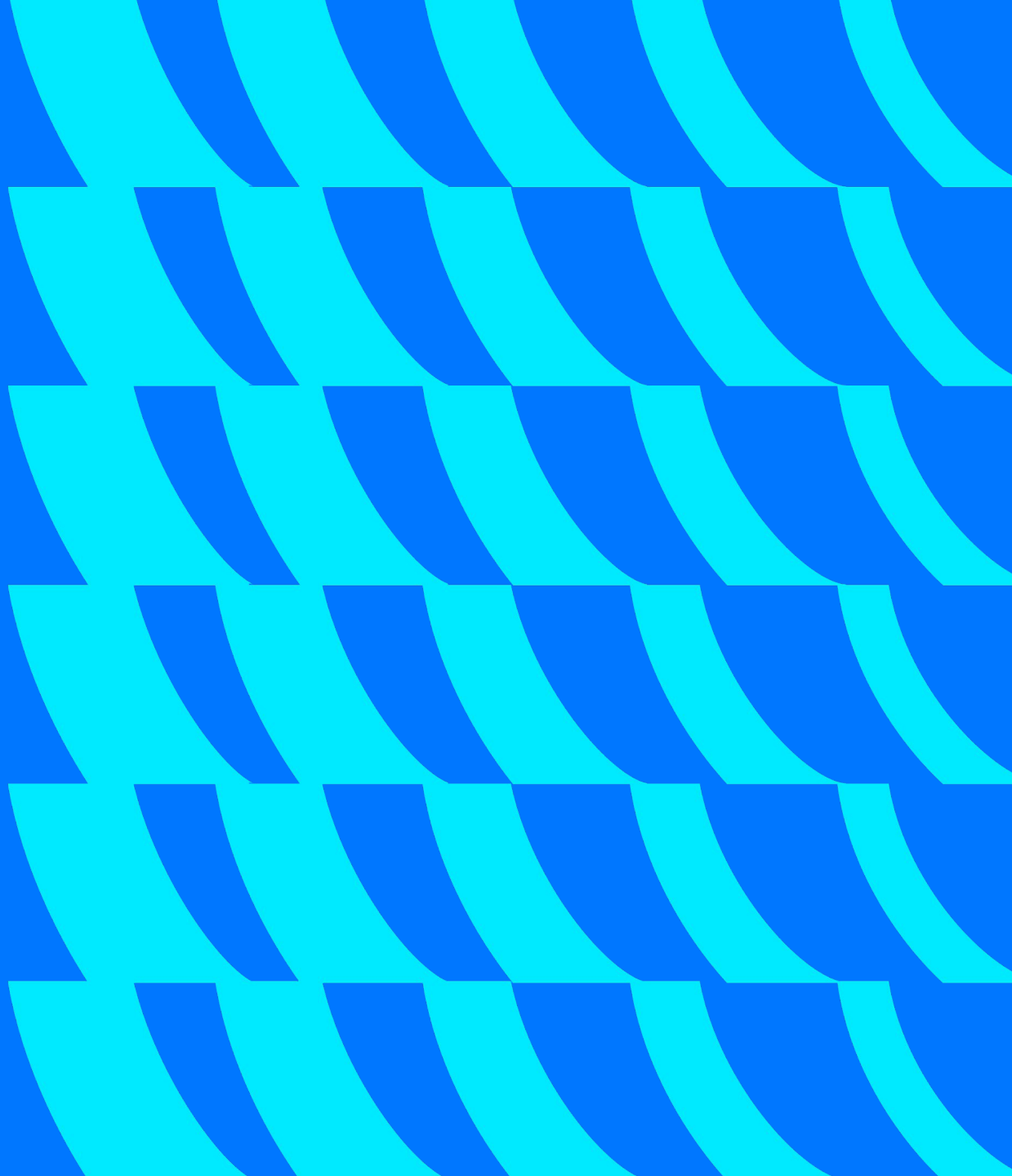
План занятия

1. Еще немного о миграциях
2. Еще немного про админку
3. Методы модели
4. Расширение модели пользователя
5. Возможности ORM



Миграции

доставляем изменения в базу



Еще немного про миграции

```
# Data Migration
# python manage.py makemigrations --empty yourappname
from django.db import migrations
def do_smth():
    ...
class Migration(migrations.Migration):
    dependencies = [
        ('yourappname', '0001_initial'),
    ]
    operations = [
        migrations.RunPython(do_smth),
    ]
```

Еще немного про миграции

```
def do_smth(apps, shema_editor):
    SomeModel = apps.get_model('some_app', 'SomeModel')
```

Админка

делаем интерфейс для моделей

Django Admin

```
# chats.admin.py

from django.contrib import admin
from chats.models import Chat

class ChatAdmin(admin.ModelAdmin):
    list_display = ('id', 'title')
    list_filter = ('is_active',)

admin.site.register(Chat, ChatAdmin)

# дока https://docs.djangoproject.com/en/3.2/ref/contrib/admin/
```

Методы модели

```
# chats.models.py

from django.db import models

class Chat(models.Model):
    title = models.CharField('Название', max_length=50)
    sorting_key = models.IntegerField('Ключ сортировки', default=0)

    def get_absolute_url(self):
        reverse('chat', kwargs={'chat_id': self.id})

    def __str__(self):
        return self.title

    class Meta:
        ordering = ['-sorting_key']
        verbose_name = 'Чат'
        verbose_name_plural = 'Чаты'
```

Расширение AbstractUser

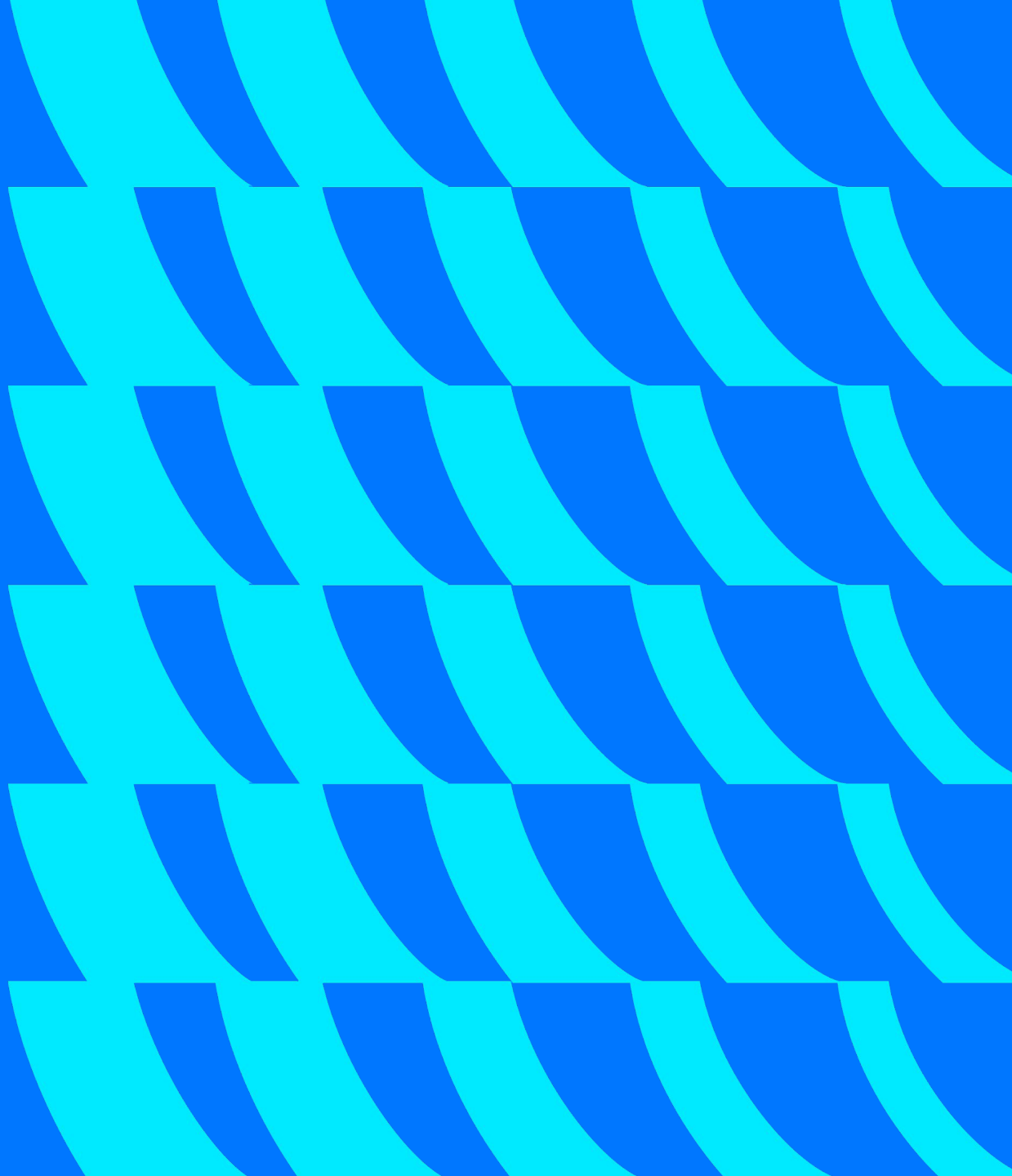
```
from django.db import models
from django.contrib.auth.models import AbstractUser


class User(AbstractUser):
    bio = models.TextField('Биография', max_length=500, blank=True)
    location = models.CharField('Город', max_length=30, blank=True)
    birthday = models.DateField('Дата рождения', null=True, blank=True)


# в settings.py добавить AUTH_USER_MODEL = 'users.YourUserModel'
```

ORM

Object-Relational Mapping



ORM. Создание объектов

```
# создание объекта без связей

post = Post()
post.title = 'Test'
post.save()

post2 = Post.objects.create(title='Test2')


# создание объекта со связями

post3 = Post.objects.create(title='Test3')
post3.category_id = 2
post3.save()

category = Category.objects.create(title='Test category')
post4 = Post.objects.create(title='Test 4', category=category)
```

Связи ManyToMany

```
post = Post.objects.create(title='Test2')
tag = Tag.objects.create(slug='some_tag')
post.tags.add(tag)
```

Получение объектов из БД

```
# по ключу

try:
    post = Post.objects.get(id=5)
except Post.DoesNotExist:
    post = None

# по другому полю

try:
    post = Post.objects.get(title='Python')
except MultipleObjectsReturned:
    post = None
```

Фильтрация объектов

```
all_posts = Post.objects.all()
first_three = Post.objects.all()[:3]

some_category = Categoty.objects.get(id=1)

category_posts = Post.objects.filter(category=some_category)
category_posts = Post.objects.filter(category_id=1)


css_posts = Post.objects.filter(title__contains='css')
css_posts = css_posts.order_by('-rating')
css_posts = css_posts[10:20]
```

Обновление и удаление

```
from django.utils import timezone


posts = Post.objects.all()
posts.update(updated_at=timezone.now())

posts.delete()
```

Queryset

Queryset

QuerySet - объекты, представляющие собой запрос к базе данных (не результаты)

QuerySet - ленивые объекты

Chaining

```
posts = Post.objects.all()

posts = posts.filter(title__contains='CSS')

posts = posts.exclude(id=21)

posts = posts.order_by('-rating')

posts = posts.reverse() # [ QuerySet ]

posts = posts[:3] # [ POST ]
```

Методы Queryset (chaining)

`filter`, `exclude` - фильтрация, WHERE в SQL

`order_by` - сортировка

`annotate` - выбор агрегатов, в SQL - JOIN и GROUP_BY

`values` - выбор отдельных колонок, а не объектов

`values_list` - то же, только без названия колонок

`distinct` - выбор уникальных значений

`select_related`, `prefetch_related` - выборка из нескольких таблиц

Методы Queryset (результат)

`create` - создание нового объекта

`update` - обновление всех подходящих объектов

`delete` - удаление одного или нескольких объектов

`get_or_create` - выборка объекта или его создание

`count` - выборка количества COUNT(*)

Синтаксис условий в filter и exclude

```
field=value - точное совпадение

field__contains=value - суффикс оператора LIKE

field__isnull, field__gt, field__lte

relation__field=value - условие по связанной таблице

category__title__contains='Python'

# Названия полей и таблиц не могут содержать __
```

ModelManager

В модели содержатся методы для работы с одним объектом (одной строкой).

В **ModelManager** содержатся объекты для работы с множеством объектов.

ModelManager по умолчанию содержит все те же методы, что и QuerySet и используется для создания QuerySet объектов, связанных с моделью.

Кастомный ModelManager

```
class PostManager(models.Manager):
    def best_posts(self):
       return self.filter(rating__gte=50)

    def published(self):
        return self.filter(status=Post.IS_PUBLISHED)

class Post(models.Model):
    title = ...
    ...
    objects = PostManager()
)
```

Методы RelatedManager

create(**kwargs) - создание новой категории, связанной с постом

add(category) - привязка существующей категории к посту

remove(category) - отвязка

clear() - очистка списка категорий у текущего поста

Что еще?

```
Post.objects.get()

Post.objects.first() # обычно после filter

Post.objects.last() # обычно после filter

Post.objects.none()

Post.objects.filter(id=2).exists()
```

Усложняем запросы

```
from django.db.models import Count
from django.db.models import Max
from django.db.models.functions import Length

posts = Post.objects.annotate(Count('tags')) # tags__count
posts = Post.objects.all().annotate(tag_count=Count('tags'))

posts = posts.filter(tag_count__gte=3)

posts = Post.objects.all().annotate(length=Length('title'))

users = User.objects.all().aggregate(age_max=Max(age))
```

Почитать

https://docs.djangoproject.com/en/3.2/misc/design-philosophies/

https://docs.djangoproject.com/en/3.2/topics/db/models/

https://docs.djangoproject.com/en/3.2/ref/models/querysets/

Домашнее задание

1. Создать свою модель пользователя
2. Переписать вьюхи так, чтобы создавались и отдавались реальные объекты (CRUD)
3. Подключить все созданные модели в админку (используем verbose name)

Не забудьте оставить отзыв на портале

Спасибо
за внимание

образование